BiasMix-Finance: Post-Generation KYC Guardrails for LLM Portfolio Advice

Anonymous authors
Paper under double-blind review

Abstract

Large language models (LLMs) can generate plausible-sounding ETF portfolios while silently violating basic KYC-style constraints on risk, fees, and diversification. We study a model-agnostic, asset-agnostic *post-generation guardrail* pipeline: (i) enforce a strict JSON allocation schema, (ii) validate allocations against numeric caps, and (iii) when violations occur, *deterministically* project the output to the nearest feasible portfolio via a convex quadratic program (QCQP). We introduce **BiasMix-Finance (Mini)**, a compact stress-test benchmark for *constrained decision-making under biased LLM generations*, with a 16-ETF universe, three investor profiles, and eight bias prompts. Across three models and three inference modes (direct, critique, self-consistency), first-pass generations violate at least one cap in **47.6–85.7%** of test cases (**67.2%** pooled), but the convex projection layer reduces *final feasibility violations to* **0%** while requiring only a *small correction distance* (test pooled median $D = \|w^* - w_0\|_2 = 0.066$), indicating that the guardrail typically preserves the intent of the original allocation. We report violation rates and correction distances with confidence intervals, and paired model comparisons with multiple-testing correction. To support reproducibility, we will release the dataset, prompts, caps, and code in an anonymous repository (URL withheld for double-blind review).

1 Introduction

Large language models (LLMs) are increasingly used to generate financial summaries, recommendations, and portfolio allocations (Wu et al., 2023; Yang et al., 2023). However, in regulated domains the output must satisfy *hard* suitability and diversification constraints (e.g., concentration limits, fee ceilings, and risk caps) that resemble KYC/suitability guardrails (Jagannathan & Ma, 2003; Boyd & Vandenberghe, 2004). Prompting and reasoning-mode variations (e.g., critique, self-consistency) are inherently probabilistic and cannot guarantee constraint satisfaction on every run (Wei et al., 2022; Wang et al., 2022; Yao et al., 2023). This motivates **post-generation enforcement**: we treat the LLM output as a draft allocation and apply deterministic verification and repair to ensure compliance (Rao et al., 2023; Meta, 2023; Ribeiro et al., 2020; Dror et al., 2018). Crucially, our goal is *not* to solve a classical portfolio-optimization problem (e.g., maximizing risk-adjusted return under constraints) (Markowitz, 1952; Black & Litterman, 1992; Goldfarb & Iyengar, 2003). Instead, we study an *auditing and repair* problem: how to reliably detect and minimally correct *irrational or constraint-violating* allocations produced by an LLM, while preserving the model's intended allocation structure as much as possible (Ribeiro et al., 2020; Bender et al., 2021; Raji et al., 2020). More broadly, this setting instantiates a common *alignment and robustness* problem: an LLM produces a plausible *decision* artifact, but correctness depends on satisfying external constraints that must be verified by deterministic checks (Ouyang et al., 2022; Bai et al., 2022). Because these constraints are externally specified and mechanically checkable, the task is a natural testbed for *post-hoc verification and repair* pipelines that are model-agnostic and transferable to other constrained decision-making domains (e.g., resource allocation, scheduling, or policy compliance). We study a controlled portfolio-allocation setting: given an ETF universe and a scenario describing (i) a risk profile with caps and (ii) a bias-inducing context (e.g., "small-cap hype"), the LLM emits portfolio weights under a strict JSON schema. A key challenge is that biased or preference-shaping contexts can systematically push generations toward concentrated, high-volatility, or otherwise non-compliant allocations. To evaluate guardrails under these realistic failure modes, we adopt a **bias-induction**

methodology: we programmatically generate scenarios that combine standardized risk profiles with bias "recipes" (e.g., anchoring on a sector, FOMO tilts, inertia, fee neglect), producing a stress-test suite for constraint-violating drafts. We then measure constraint violations *before* repair and the magnitude of correction required *after* repair. Our goal is to quantify how much prompt reasoning modes reduce violations *prior* to enforcement, and how much post-hoc guardrails must change the portfolio to make it compliant. In this sense, we treat BiasMix-Finance (Mini) as a stress-test benchmark for *constrained decision-making under biased generations*, complementing broader evaluation efforts in safe and reliable LLM deployment. To verify that BiasMix prompts elicit the intended behavioral tilts, Appendix Table 6 shows representative first-pass LLM outputs (held-out test) for each bias recipe.

Contributions.

- **Bias-inducing evaluation suite for financial guardrails.** We construct 72 scenario instances (train/dev/test) that combine three risk profiles with eight behavioral "bias recipes" and three random seeds, over a fixed 16-ETF universe. This controlled design isolates how different prompts and models respond to the same constraints and bias contexts, and makes bias induction itself a reproducible evaluation primitive for constrained decision-making under LLM outputs.
- **Post-generation constraint enforcement for LLM portfolio drafts.** We formalize the feasible set induced by volatility, fee, HHI concentration, and max single-asset/sector caps, and compute the nearest feasible portfolio via a constrained quadratic program, solved with standard convex optimization tools (Boyd & Vandenberghe, 2004; Diamond & Boyd, 2016; Stellato et al., 2020).
- **Reproducible statistical evaluation across splits, models, and modes.** We report per-cap violation rates with Wilson 95% confidence intervals (Wilson, 1927), correction-distance and metric-delta confidence intervals via bootstrap (Efron & Tibshirani, 1993), and paired model comparisons on correction distance with Wilcoxon tests plus BH-FDR correction (Wilcoxon, 1945; Benjamini & Hochberg, 1995).

2 Problem Setup

Universe. Let n denote the number of ETFs in the universe (here $n = 16$) and let $w \in \mathbb{R}^n$ denote portfolio weights.

Scenario. Each scenario provides (i) a risk profile with caps and (ii) a bias context string intended to nudge the model toward potentially non-compliant allocations. Caps include an annualized volatility ceiling σ_{cap}, a weighted-average expense ratio ceiling f_{cap}, a concentration ceiling h_{cap} based on the Herfindahl–Hirschman Index (HHI) (Herfindahl, 1950; Hirschman, 1964), and max single-asset and max single-sector weight caps. Such constraints are standard in constrained portfolio construction and are commonly studied as practical overlays on mean–variance style allocations (Markowitz, 1952; Jagannathan & Ma, 2003).

Model output. The LLM emits a draft portfolio w_0 (JSON with nonnegative weights that sum to 1). We compute summary metrics: risk $\sigma(w) = \sqrt{w^\top \Sigma w}$ (Markowitz, 1952), fee $\text{WAER}(w) = \sum_i w_i \cdot \text{fee}_i$, and concentration $\text{HHI}(w) = \sum_i w_i^2$.

Feasible set. Let $\mathcal{C}$ be the set of portfolios satisfying all caps. A draft is *violating* if $w_0 \notin \mathcal{C}$.

Post-generation enforcement. When w_0 violates any cap, we compute the nearest feasible portfolio

$$w^* = \arg\min_w \|w - w_0\|_2^2 \quad \text{s.t.} \quad w \in \mathcal{C}, \tag{1}$$

and define the correction distance as $D = \|w^* - w_0\|_2$ (Euclidean/L2 distance). Projection operators are a standard way to enforce feasibility under convex constraints (Boyd & Vandenberghe, 2004; Duchi et al., 2008), and related portfolio work often uses robust or constrained formulations to stabilize allocations under estimation error (Black & Litterman, 1992; Goldfarb & Iyengar, 2003; Ben-Tal et al., 2009).

3 RELATED WORK

LLMs for finance. Domain-specific financial LLMs such as BloombergGPT (Wu et al., 2023) and open, community-driven efforts like FinGPT (Yang et al., 2023) demonstrate strong capability on financial NLP tasks. Our focus differs: we study *hard constraint compliance* for portfolio-weight outputs rather than text generation accuracy.

Guardrails, auditing, and post-hoc verification. Safety and controllability toolkits (e.g., NeMo Guardrails (Rao et al., 2023)), safety classifiers (e.g., Llama Guard (Meta, 2023)), and instruction/alignment methods such as RLHF and Constitutional AI (Ouyang et al., 2022; Bai et al., 2022) primarily aim to filter unsafe content or enforce conversational policies. A complementary line of work emphasizes *auditing* and *verification* of LLM outputs, especially in high-stakes settings: rather than trusting a single generation, systems instrument the generation process with structured outputs, validation checks, and feedback loops (Ribeiro et al., 2020; Dror et al., 2018). Constrained generation methods provide decoding-time guarantees by enforcing lexical or structural constraints during generation (Hokamp & Liu, 2017). In contrast to content safety, our setting requires satisfying quantitative constraints that are naturally expressed as convex restrictions over portfolio weights; this motivates optimization-based post-processing as an auditable repair layer rather than purely text-level controls (Boyd & Vandenberghe, 2004).

Structured generation and constrained outputs. A practical prerequisite for auditing is that model outputs are machine-checkable. Prior work on structured outputs (e.g., JSON/function-style interfaces) and constrained generation motivates enforcing schemas so downstream validators can reliably parse and evaluate outputs. Our pipeline adopts this principle by requiring a strict JSON schema for allocations, enabling deterministic constraint checking and repair.

Constrained portfolio optimization. Classical mean–variance optimization (Markowitz, 1952) and later robust/regularized variants (Jagannathan & Ma, 2003) motivate using covariance structure and explicit constraints in portfolio construction. Our work uses these tools as a *repair operator* applied to an LLM draft, aligning with the broader view that LLM outputs may require deterministic verification in high-stakes domains. Unlike traditional optimization, we do not assume the portfolio is being constructed from first principles to maximize an objective; instead, we minimize deviation from the model's proposal while enforcing KYC-style caps.

Reasoning modes and sampling. Prompting strategies such as chain-of-thought (Wei et al., 2022) and self-consistency (Wang et al., 2022) can improve reasoning reliability but remain stochastic. We therefore treat mode comparisons as empirical outcomes and quantify uncertainty with confidence intervals and paired tests, and we position deterministic post-generation verification/repair as the mechanism that guarantees compliance regardless of prompting mode.

4 METHODOLOGY

Figure 1 summarizes the end-to-end BiasMix-Finance pipeline from scenario construction to deterministic feasibility via projection (Boyd & Vandenberghe, 2004; Ribeiro et al., 2020).

4.1 UNIVERSE AND SCENARIOS

We construct a synthetic-but-controlled scenario suite over a fixed ETF universe of $n = 16$ liquid funds: `SPY, VEA, VWO, VGT, XLE, XLF, XLV, XLY, XLP, XLI, XLRE, IWM, AGG, LQD, IEF, GLD`. Each scenario specifies an *as-of* date (we use `2025-09-30`), a risk profile (Conservative/Moderate/Aggressive), and a *BiasMix* recipe that nudges the model toward a known behavioral bias (e.g., "anchor tech", "FOMO energy", "small-cap hype"). Recipes are expressed as short natural-language preferences (not hard constraints) so that first-pass outputs can be either feasible or infeasible. This design parallels behavioral stress testing in NLP evaluation: the goal is to systematically elicit failure modes under controlled perturbations (Ribeiro et al., 2020). We generate 72 scenarios across 3 profiles and 8 bias recipes, with *three* random seeds per (profile, recipe) cell; split sizes are train/dev/test = 36/15/21, balanced across the three risk profiles

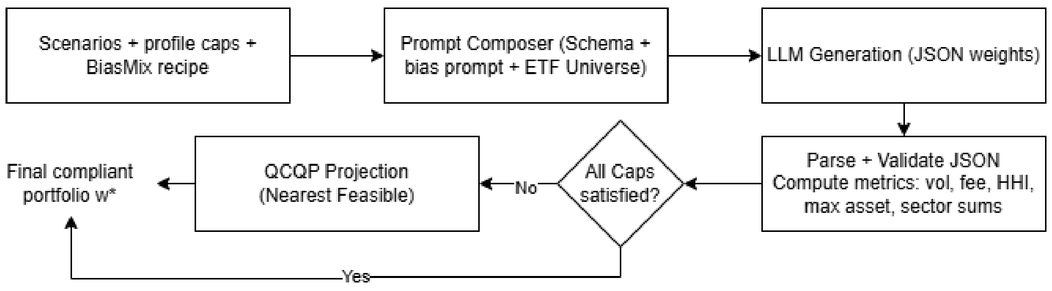
Scenarios + profile caps + BiasMix recipe
Prompt Composer (Schema + bias prompt + ETF Universe)
LLM Generation (JSON weights)
Parse + Validate JSON Compute metrics: vol, fee, HHI, max asset, sector sums
All Caps satisfied?
No
QCQP Projection (Nearest Feasible)
Final compliant portfolio w*
Yes

Figure 1: **End-to-end pipeline.** BiasMix scenario + bias recipe are composed into a structured prompt. The LLM generates draft weights (JSON). We validate JSON and compute portfolio metrics against hard KYC caps. If any constraint fails, we repair via convex projection (QCQP) to the nearest feasible portfolio; otherwise we accept directly.

Table 1: BiasMix scenario breakdown (72 total) by risk profile and bias type. Each (profile, bias) cell has 3 scenarios (3 random seeds).

Profile	**Anchor tech**	**Default inertia**	**EM tilt**	**FOMO energy**	**Fee neglect**	**Gold craze**	**Small–cap hype**	**US–only bias**	**Total**
Conservative	3	3	3	3	3	3	3	3	24
Moderate	3	3	3	3	3	3	3	3	24
Aggressive	3	3	3	3	3	3	3	3	24
Total	9	9	9	9	9	9	9	9	72

(each profile contributes 12/5/7 to train/dev/test). We use train only for prompt and hyperparameter iteration (e.g., mode temperatures, self-consistency sample size K, and projection margins). Dev a one-time check that the tuned pipeline behaves similarly on unseen data. We report results on test without further tuning. Table 1 summarizes the dataset.

4.2 Risk profiles and hard caps

All caps are defined over the portfolio functions in Section 2. For each risk profile we fix a cap vector $\theta = (\sigma_{\text{cap}}, f_{\text{cap}}, h_{\text{cap}}, a_{\max}, s_{\max})$ covering annualized volatility, weighted-average expense ratio (WAER), concentration (HHI), and single-asset / single-sector limits. Table 2 lists the numerical values used in all experiments.

Cap rationale The caps in Table 2 are chosen as representative KYC-style suitability heuristics (not jurisdiction-specific rules); detailed justification and threshold sensitivity are in Appendix B.1.

4.3 Train/dev/test splits

Although we do not train the language models, we use splits to avoid *evaluation leakage* from repeated prompt and guardrail tuning on the same scenarios. We stratify scenarios by (profile, recipe) and assign them to **train** (used only to finalize prompts and guardrail solver settings), **dev** (a one-time check that the tuned pipeline behaves similarly on unseen scenarios), and **test** (the final reported results).

4.4 Guardrail pipeline

Given a scenario, the model outputs portfolio weights w_0 under a strict JSON schema. We validate w_0; if any constraint in $\mathcal{C}(\theta)$ is violated, we compute a nearest-feasible projection:

$$w^* \;=\; \arg\min_{w \in \mathcal{C}(\theta)} \; \|w - w_0\|_2^2 \;+\; \lambda\, w^\top \Sigma w, \tag{2}$$

Table 2: Risk-profile caps used throughout. σ_{cap} is annualized volatility, WAER is weighted-average expense ratio, HHI is concentration, $a_{\max}$ is max single-asset weight, and $s_{\max}$ is max single-sector weight.

Profile	σ_{cap}	f_{cap} **(WAER)**	h_{cap} **(HHI)**	$a_{\max}$	$s_{\max}$
Conservative	0.06	0.0020	0.12	0.25	0.30
Moderate	0.10	0.0030	0.18	0.35	0.40
Aggressive	0.14	0.0040	0.25	0.45	0.50

where feasible set $\mathcal{C}$ is induced by the hard caps:

$$\mathcal{C} = \left\{ w \in \mathbb{R}^n : \begin{array}{c} \mathbf{1}^\top w = 1,\ w \geq 0, \\ \sigma(w) \leq \sigma_{\text{cap}},\ \text{WAER}(w) \leq f_{\text{cap}}, \\ \text{HHI}(w) \leq h_{\text{cap}},\ \max_i w_i \leq a_{\text{cap}}, \\ \text{sector\_sum}(w) \leq s_{\text{cap}} \end{array} \right\} \tag{3}$$

$\lambda = 2 \times 10^{-3}$ is a small variance regularizer that discourages solutions sitting exactly on the risk boundary. We report the *correction distance* $D = \|w^* - w_0\|_2$ (Euclidean norm). We solve the projection as a convex QCQP with standard solvers; numerical stability choices (PSD jitter), solver tolerances, fallback strategy, and robustness ablations are reported in the Appendix D

4.5 Models and providers

We evaluate three LLM backends: (i) *Gemini 2.5 Flash* (`gemini-2.5-flash`) via the Google GenAI Python SDK, (ii) *GPT-5 nano* (`gpt-5-nano`) via the OpenAI Chat Completions API, and (iii) an open-weight model served through an OpenAI-compatible endpoint, *Llama 3.3 70B Instruct Turbo* (`meta-llama/Llama-3.3-70B-Instruct-Turbo`).

4.6 Inference Strategies (Modes)

The unit of analysis is a *scenario* evaluated under a (model, mode) configuration. We evaluate the models across three strategies: (i) **Direct** generation, where the model is prompted to produce the target JSON output in a single shot; (ii) **Critique**, a multi-turn approach where the model generates an internal draft and critique before emitting a corrected JSON; and (iii) **Self-consistency (SC)**, where $K = 5$ candidates are sampled, and the candidate with the lowest pre-projection violation is selected. All modes share the same post-generation projection step when constraints are violated and enforce strict JSON outputs with up to R=3 retries on parse failure; once parsed, weights are validated and projected if needed. Full decoding and parsing details are in the Appendix E.

5 Experimental Protocol and Statistical Framework

5.1 Primary endpoints

We pre-register two primary endpoints: (1) **First-pass violation rate.** For each cap (and the any-cap aggregate), we mark a violation on w_0 if the corresponding constraint in $\mathcal{C}$ is not satisfied (e.g., $\sigma(w_0) > \sigma_{\text{cap}}$).; and (2) **correction distance** $D = \|w^* - w_0\|_2$ (Euclidean distance) measuring how much the guardrail must change the model output to satisfy hard caps.

5.2 Secondary endpoints

We report before/after deltas in volatility σ, fee burden (WAER), and concentration (HHI), as well as parse-failure rate and end-to-end success (valid JSON + post-generation feasibility). **(i) Parse failure rate.** A scenario is a parse failure if all R=3 retries fail strict JSON parsing. **(iii) Final feasibility rate.** A scenario is finally feasible if $w^* \in \mathcal{C}$ (all caps satisfied after projection). **(iv) End-to-end success.** A scenario is end-to-end successful if it parses *and* is finally feasible.

5.3 Confidence intervals

For violation rates (proportions), we compute Wilson 95% confidence intervals (Wilson, 1927). For D and metric deltas, we compute bootstrap 95% confidence intervals by resampling scenarios with replacement (10,000 resamples) (Efron & Tibshirani, 1993).

5.4 Model comparisons and multiple testing

To compare models on D under the same scenarios, we use paired Wilcoxon signed-rank tests (Wilcoxon, 1945; Demšar, 2006). We correct for multiple pairwise tests using Benjamini–Hochberg false discovery rate (FDR) control at $\alpha = 0.10$ (Benjamini & Hochberg, 1995) Appendix F.

5.5 Split-wise reporting

Train is used only to finalize prompts and solver settings. Dev is reassure settings. We do not re-tune based on test outcomes, instead we report test metrics (with confidence intervals) as our main evidence of generalization across unseen scenarios and paired comparisons as the main empirical results.

6 Results

6.1 First-pass violations on held-out test

Table 3 and Figure 2 report pooled first-pass violation rates on the test split. Critique and self-consistency tend to reduce violations relative to direct prompting, but do not guarantee compliance. Violation rates are heterogeneous across BiasMix recipes and profiles; Appendix Figs. 4–6 visualize test-split violation heatmaps by bias type, profile, and model (for each prompting mode). Interpreting the magnitudes, prompting alone remains unreliable under hard KYC-style caps: even the best test setting (Gemini+SC) violates in nearly half of cases (0.476), while the worst (Gemini+direct) violates most of the time (0.857). Self-consistency helps most when the feasible region is larger (Aggressive drops from 0.476 to 0.048), but tight regimes remain difficult: Conservative portfolios violate on all test scenarios across models/modes (1.0), motivating a deterministic verify-and-repair layer.

6.2 Projection yields high final feasibility

Most failures arise at parse time (invalid JSON), not at constraint time. On the held-out test split, with strict schema prompting and up to $R = 3$ retries, we observe *zero parse failures* and *100% post-projection feasibility* across all models and modes (Wilson 95% CIs: parse-fail $[0, 0.155]$, final pass $[0.845, 1.0]$; full table in Appendix C.1). Thus, reliability hinges primarily on structured output; once weights are parsed, solver-based repair enforces constraints deterministically while preserving the LLM as an intent generator.

6.3 How much correction is required?

Table 4 and Figure 3 quantify the adjustment required to reach feasibility. Lower D indicates the model proposed a near-feasible portfolio. We interpret D as a faithfulness signal: small D means projection only nudges the draft onto the feasible set, while large D means constraints substantially override the draft. Consistent with tightness, corrections are smallest for Aggressive (often near zero, especially with SC), moderate for Moderate (roughly 0.018–0.185), and largest for Conservative (roughly 0.220–0.462), aligning with its near-certain first-pass violations. In deployment, large D can be surfaced as "constraint-forced rebalancing," while small D supports the claim that guardrails can guarantee compliance without materially changing near-feasible drafts. Beyond L2 distance, intent is also preserved at a coarser granularity: Appendix Figs. 7–9 plot draft vs. projected *sector* allocations on the test split, with points concentrated near the diagonal. On the held-out test split (pairs=21), differences are significant in *direct* and *critique* but not *SC*: gpt-5-nano vs meta-llama/Llama-3.3-70B-Instruct-Turbo is significant in direct (p_{FDR}=0.0292) and critique

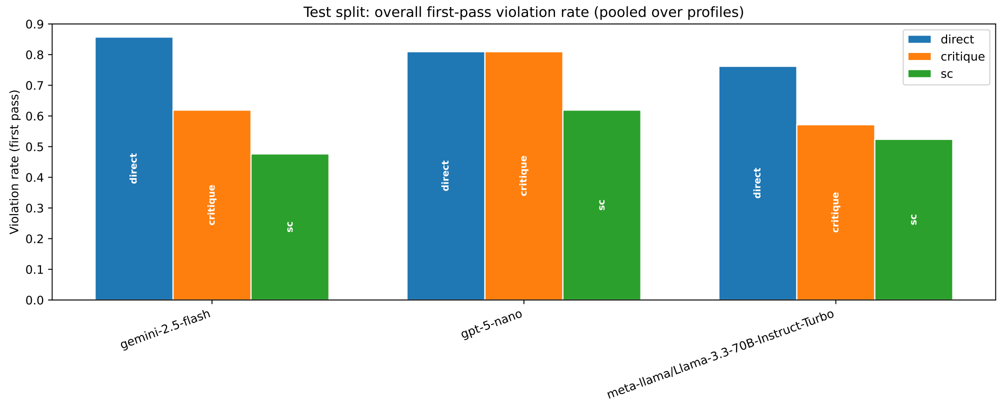

Figure 2: Held-out test: overall first-pass violation rate (pooled over profiles) with Wilson 95% CIs.

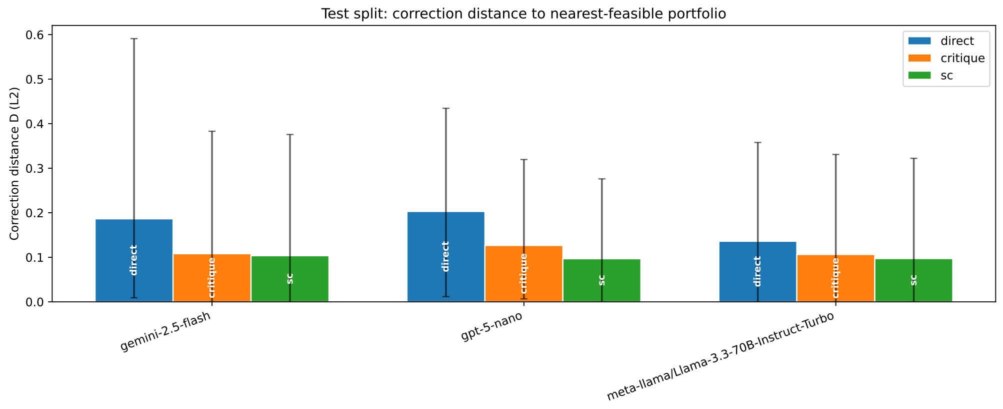

Figure 3: Held-out test: correction distance $D = \|w^* - w_0\|_2$ (profile-pooled) with conservative CI bands.

(p_{FDR}=0.0098), and Gemini-2.5-flash vs meta-llama/Llama-3.3-70B-Instruct-Turbo is also significant in direct (p_{FDR}=0.0669) and critique (p_{FDR}=0.0098). Full results across splits and modes are in Appendix 7.

7 Discussion, limitations, and future work

Prompting modes (direct, critique, self-consistency) can reduce first-pass violations, but they cannot guarantee satisfaction of hard numeric caps because generation remains probabilistic. In contrast, the post-generation projection step provides an auditable, model-agnostic enforcement layer that deterministically returns a feasible portfolio (when the cap set is feasible), while minimizing deviation from the model's proposal.

Scope and limitations. This study is intentionally scoped to *constraint compliance and robustness*, not portfolio optimality: we test whether LLM allocations can be made KYC-feasible under explicit caps and quantify the minimal adjustment required (correction distance D). We therefore do not model expected returns, transaction costs, taxes, or utility-based objectives. We use a fixed *16-ETF* universe as a controlled stress test; the guardrail is not tied to 16 assets and applies to any universe with fees, sector mappings, and a covariance estimate. We also do not conduct real-user or advisor-in-the-loop studies, focusing instead on an auditable enforcement mechanism for downstream workflows (Wei et al., 2022; Wang et al., 2022; Yao et al., 2023; Rao et al., 2023; Meta, 2023; Ribeiro et al., 2020; Bender et al., 2021; Raji et al., 2020). Additional discussion on universe design and generalization is provided in Appendix G.

Table 3: **Test first-pass violation rate (Wilson 95% CI).** Violations are measured on the model's raw proposal w_0 before projection; lower values indicate better prompt-level compliance, not guardrail effectiveness.

Model	Mode	n	Viol.	Violation rate (95% CI)
gemini-2.5-flash	critique	21	13	0.619 [0.409,0.792]
gemini-2.5-flash	direct	21	18	0.857 [0.654,0.950]
gemini-2.5-flash	sc	21	10	0.476 [0.283,0.676]
gpt-5-nano	critique	21	17	0.810 [0.600,0.923]
gpt-5-nano	direct	21	17	0.810 [0.600,0.923]
gpt-5-nano	sc	21	13	0.619 [0.409,0.792]
meta-llama/Llama-3.3-70B-Instruct-Turbo	critique	21	12	0.571 [0.365,0.755]
meta-llama/Llama-3.3-70B-Instruct-Turbo	direct	21	16	0.762 [0.549,0.894]
meta-llama/Llama-3.3-70B-Instruct-Turbo	sc	21	11	0.524 [0.324,0.717]

Table 4: **Test correction distance D (bootstrap 95% CI).** $D = \|w^* - w_0\|_2$ (unitless L2 distance on the simplex); lower values mean the model output was closer to feasible before projection.

Model	Mode	n	D **mean (band)**	D **median**
gemini-2.5-flash	critique	21	0.108 [0.000,0.383]	0.028
gemini-2.5-flash	direct	21	0.186 [0.009,0.591]	0.048
gemini-2.5-flash	sc	21	0.104 [0.000,0.376]	0.000
gpt-5-nano	critique	21	0.127 [0.007,0.320]	0.121
gpt-5-nano	direct	21	0.203 [0.012,0.435]	0.110
gpt-5-nano	sc	21	0.097 [0.000,0.276]	0.072
Llama-3.3-70B-Instruct	critique	21	0.106 [0.000,0.331]	0.029
Llama-3.3-70B-Instruct	direct	21	0.136 [0.000,0.358]	0.119
Llama-3.3-70B-Instruct	sc	21	0.097 [0.000,0.322]	0.028

Future work. Promising extensions include (i) scaling to larger universes and adding liquidity/turnover constraints, (ii) replacing the fixed covariance with rolling or regime-aware estimates, (iii) incorporating return-aware or utility-based objectives while preserving hard caps, and (iv) running human-evaluation studies to assess whether corrected portfolios preserve user intent and improve trust and usability.

8 Conclusion

Post-generation KYC-style guardrails offer a practical and auditable way to enforce numeric investment constraints on top of LLM-generated allocations. Across models and prompting modes, we find that a convex nearest-feasible projection can eliminate final constraint violations while keeping repairs bounded in L2 distance. More generally, the same verify-and-repair pattern is *model-agnostic* and *asset-agnostic*: any LLM that emits a structured decision can be checked against hard constraints and deterministically repaired when needed, making the approach applicable beyond finance. BiasMix-Finance (Mini) serves as a stress-test benchmark for constrained decision-making under bias, enabling controlled comparisons of reasoning modes and models under identical caps and contexts.

LLM usage disclosure. We used large language models as auxiliary tools for language polishing and limited code assistance (e.g. small code utilities and LaTeX formatting). All experimental design, implementation, data collection, analysis, results, and conclusions were produced and verified by the authors, who take full responsibility for the content of this paper.

References

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Dan Goldie, Azalia Mirhoseini, Chris McKinnon, et al. Constitutional AI: Harmlessness from AI feedback. *arXiv preprint arXiv:2212.08073*, 2022.

Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust Optimization*. Princeton University Press, 2009.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, 2021.

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

Fischer Black and Robert Litterman. Global portfolio optimization. *Financial Analysts Journal*, 48 (5):28–43, 1992.

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

Janez Demšar. Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7:1–30, 2006.

Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.

John Duchi, Shai Shalev-Shwartz, Yoram Singer, and Tushar Chandra. Efficient projections onto the ℓ_1-ball for learning in high dimensions. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, 2008.

Bradley Efron and Robert Tibshirani. *An Introduction to the Bootstrap*. Chapman & Hall/CRC, 1993.

Donald Goldfarb and Garud Iyengar. Robust portfolio selection problems. *Mathematics of Operations Research*, 28(1):1–38, 2003.

Orris C. Herfindahl. *Concentration in the steel industry*. PhD thesis, Columbia University, 1950.

Albert O. Hirschman. The paternity of an index. *The American Economic Review*, 54(5):761–762, 1964.

Chris Hokamp and Qun Liu. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.

Ravi Jagannathan and Tongshu Ma. Risk reduction in large portfolios: Why imposing the wrong constraints helps. *Journal of Finance*, 58(4):1651–1683, 2003.

Harry Markowitz. Portfolio selection. *The Journal of Finance*, 7(1):77–91, 1952.

Meta. Llama Guard: LLM-based input-output safeguard for human-AI conversations. *arXiv preprint arXiv:2312.06674*, 2023.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

Inioluwa Deborah Raji, Andrew Smart, Rebecca N. White, Margaret Mitchell, Timnit Gebru, Ben Hutchinson, Jamila Smith-Loud, Daniel Theron, and Parker Barnes. Closing the AI accountability gap: Defining an end-to-end framework for internal algorithmic auditing. In *Proceedings of the 2020 ACM Conference on Fairness, Accountability, and Transparency (FAccT)*, 2020.

Saurabh Rao, Paul Buehler, et al. NeMo Guardrails: A toolkit for controllable and safe LLM applications with programmable rails. *arXiv preprint arXiv:2310.15851*, 2023.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2020.

Bartolomeo Stellato, Goran Banjac, Paul Goulart, Alberto Bemporad, and Stephen Boyd. OSQP: An operator splitting solver for quadratic programs. *Mathematical Programming Computation*, 12:637–672, 2020.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

Frank Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

Edwin B. Wilson. Probable inference, the law of succession, and statistical inference. *Journal of the American Statistical Association*, 22(158):209–212, 1927.

Shijie Wu, Jeffrey Le, Adam Atkinson, Walid Krichene, Derek Scott, Ayan Chakraborty, et al. BloombergGPT: A large language model for finance. *arXiv preprint arXiv:2303.17564*, 2023.

Hongyang Yang, Shaozhi Zhou, et al. FinGPT: Open-source financial large language models. *arXiv preprint arXiv:2306.06031*, 2023.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

A Appendix

B Prompts, caps, and scenario template

B.1 Cap rationale and sensitivity

The cap values in Table 2 are intended to be representative of practical "KYC-style" suitability heuristics rather than jurisdiction-specific regulations. They reflect: (i) lower volatility targets for conservative investors; (ii) a low-cost fee budget typical of broad-market ETFs (e.g., 0.20–0.40% WAER); (iii) diversification via concentration control (HHI corresponds to an effective number of holdings of roughly $1/\text{HHI}$); and (iv) concentration limits that prevent over-exposure to a single fund or sector. (We report threshold sensitivity analyses in this appendix.)

B.2 Direct/Critique JSON schema

```
{"weights": {"TICKER": 0.00, "...": 0.00},
"explanation": "<=120 words"}
```

B.3 SC JSON SCHEMA

```
{
  "candidates": [{"weights": {"TICKER": 0.00, ...},
"explanation": "<=120 words"},
    ...
  ]
}
```

B.4 EXAMPLE BIASMIX PROMPTS

We include two representative BiasMix contexts (verbatim) used to induce biased first-pass allocations:

- **Anchor tech:** *Strongly prefer VGT (tech tilt).*
- **Small-cap hype:** *Strongly prefer IWM (small-cap tilt).*

B.5 SCENARIO JSONL TEMPLATE

```
{
  "scenario_id": "AF-001",
  "as_of": "2025-09-30",
  "profile": "Conservative|Moderate|Aggressive",
  "caps": {"sigma_cap":..., "fee_cap":..., "hhi_cap":...,
"max_asset":..., "max_sector":...},
  "asset_universe": ["SPY","VEA","VWO","VGT","XLE","XLF","XLV",
    "XLY","XLP","XLI","XLRE","IWM","AGG","LQD","IEF","GLD"],
  "biasmix": "<bias recipe text>",
  "seeds": [1],
  "split": "train|dev|test"
}
```

C ADDITIONAL RESULTS

C.1 TEST ROBUSTNESS AND SUCCESS RATES (TABLE 5)

C.2 QUALITATIVE BIAS PROMPT EFFECTIVENESS (LLM OUTPUT EXAMPLES)

Table 6 provides one concrete held-out test example per BiasMix recipe, showing that the first-pass draft allocations and the model's explanation reflect the intended bias (e.g., VGT/XLE/GLD/IWM tilts). We show direct prompting for brevity; critique/SC exhibit similar thematic emphasis but are omitted due to space.

C.3 BIAS ANALYSIS ON HELD-OUT TEST (VIOLATION HEATMAPS)

To inspect bias-specific failure modes, we compute first-pass violation rates on the *test split* for each (profile, bias type) cell, shown separately by model and prompting mode in Figs. 4–6. Cells with no test scenarios for a given (profile, bias) combination are left blank.

Table 5: **Test robustness and success rates (Wilson 95% CI).** Parse-fail counts a scenario only if all R=3 retries fail strict JSON parsing. Final feasibility is the post-projection pass rate ($w^* \in \mathcal{C}$). End-to-end success requires both parsing and final feasibility.

Model	**Mode**	n	**Parse fails**	**Parse-fail (95% CI)**	**Final pass (95% CI)**
gemini-2.5-flash	critique	21	0	0.000 [0.000, 0.155]	1.000 [0.845, 1.000]
gemini-2.5-flash	direct	21	0	0.000 [0.000, 0.155]	1.000 [0.845, 1.000]
gemini-2.5-flash	sc	21	0	0.000 [0.000, 0.155]	1.000 [0.845, 1.000]
gpt-5-nano	critique	21	0	0.000 [0.000, 0.155]	1.000 [0.845, 1.000]
gpt-5-nano	direct	21	0	0.000 [0.000, 0.155]	1.000 [0.845, 1.000]
gpt-5-nano	sc	21	0	0.000 [0.000, 0.155]	1.000 [0.845, 1.000]
Llama-3.3-70B	critique	21	0	0.000 [0.000, 0.155]	1.000 [0.845, 1.000]
Llama-3.3-70B	direct	21	0	0.000 [0.000, 0.155]	1.000 [0.845, 1.000]
Llama-3.3-70B	sc	21	0	0.000 [0.000, 0.155]	1.000 [0.845, 1.000]

Table 6: **Bias prompt effectiveness on held-out test:** representative first-pass LLM draft allocations and short explanation excerpts for each BiasMix recipe (direct prompting shown for compactness).

Bias recipe	**Scenario**	**Draft weights (excerpt)**	**Explanation snippet (verbatim)**
Anchor tech	AF-051 (Agg.)	VGT 45%, SPY 30%, VEA 15%, VWO 10%	"strong tech tilt via VGT (45%), aligning with the 'Anchor tech' bias"
FOMO energy	AF-004 (Cons.)	XLE 10%, AGG 40%, LQD 20%, IEF 17%	"strategic tilt towards XLE (energy) . . . aligning with the FOMO energy bias"
Small-cap hype	AF-061 (Agg.)	IWM 25%, SPY 30%, VEA 15%, VWO 10%	"prioritizes . . . small-cap exposure via IWM, aligning with the 'small-cap hype' bias"
Gold craze	AF-047 (Mod.)	GLD 20%, SPY 20%, AGG 20%, VEA 15%	"a significant gold tilt (GLD) as requested"
EM tilt	AF-060 (Agg.)	VWO 30%, SPY 30%, VEA 20%, VGT 10%	"strong EM tilt via VWO, complemented by VEA"
US-only bias	AF-020 (Cons.)	AGG 60%, SPY 15%, LQD 10%, IEF 10%, XLP 5%	"US equity exposure is limited to SPY . . . aligning with the US-only bias"
Default inertia	AF-009 (Cons.)	AGG 100%	"default inertia to favor AGG materially . . . a 100% allocation to AGG"
Fee neglect	AF-066 (Agg.)	SPY 35%, VEA 20%, VWO 15%, IWM 10%, AGG 10%	"diversified . . . and US small-cap (IWM) . . . respecting the fee neglect bias"

C.4 SECTOR-LEVEL INTENT PRESERVATION (BEFORE/AFTER)

To assess intent preservation beyond the L2 correction distance, we compare *sector-sum* allocations before and after projection on the held-out test split. Each point in Figs. 7–9 corresponds to a (scenario, sector) pair, plotting the draft sector weight (x-axis) against the post-projection sector weight (y-axis). Points close to the $y = x$ line indicate that the projection repair preserves the model's sector-level intent while enforcing hard caps.

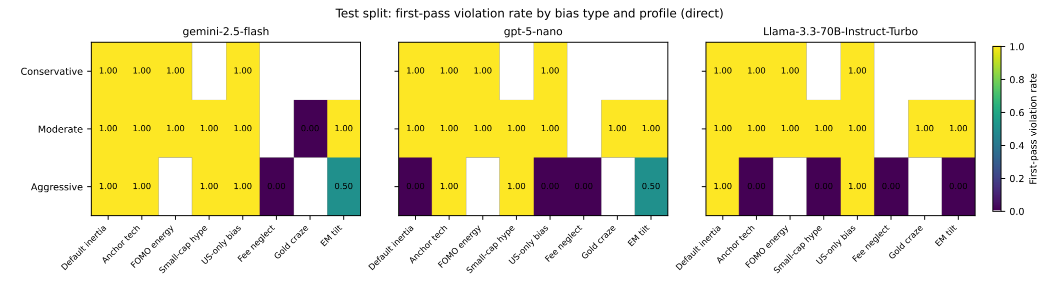

Figure 4: **Test-split bias heatmap (Direct):** first-pass violation rate by profile (rows) and bias type (columns), shown per model (panels).

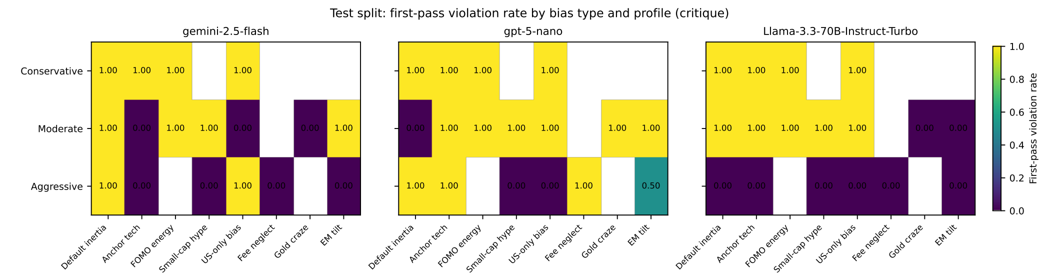

Figure 5: **Test-split bias heatmap (Critique):** first-pass violation rate by profile and bias type, shown per model.

C.5 Paired model comparisons (Wilcoxon on correction distance D)

Table 7 reports paired Wilcoxon signed-rank tests comparing models on the correction distance D (only scenarios where both models produced a valid run are paired). We report the Wilcoxon statistic W, the unadjusted p-value, the effect size r, and the Benjamini–Hochberg FDR-adjusted p-value (p_{FDR}). "Reject" indicates rejection at FDR $\alpha = 0.10$.

Interpretation Across splits, paired Wilcoxon tests on correction distance D show that *direct* and *critique* modes often yield statistically distinguishable D distributions between certain model pairs after BH-FDR correction (notably comparisons involving `meta-llama/Llama-3.3-70B-Instruct-Turbo`). In contrast, *sc* mode exhibits fewer significant differences across model pairs (most $p_{\mathrm{FDR}} \geq 0.10$), suggesting that the self-consistency selection tends to reduce cross-model variability in the magnitude of post-generation corrections. We report these as empirical outcomes rather than enforcing an a priori ordering between modes.

D Solver Projection Implementation

D.1 Solver Settings

Convexity Because $\Sigma \succeq 0$ and the constraints in $\mathcal{C}(\theta)$ are convex (simplex, box, linear sector sums, and convex quadratic bounds on volatility and HHI), Eq. equation 2 is a convex QCQP. We implement it in CVXPY with a PSD "jitter" $\Sigma \leftarrow \Sigma + 10^{-10} I$ for numerical stability. In practice, the quadratic volatility/HHI caps induce a conic form, so we solve the QCQP with SCS (`eps=1e-5`, `max_iters=30000`). For robustness, we optionally attempt OSQP when CVXPY canonicalizes a given instance to a pure QP (no quadratic caps), using `eps_abs=eps_rel=1e-7`, `max_iter=200000`, and `polish=True`; otherwise (or if a solver returns a non-optimal status) we fall back to SCS. This retry logic improves robustness across models and scenarios.

Robustness to solver settings Appendix D.2 reports a small sensitivity study over solver tolerances and iteration limits; feasibility and correction distance remain stable.

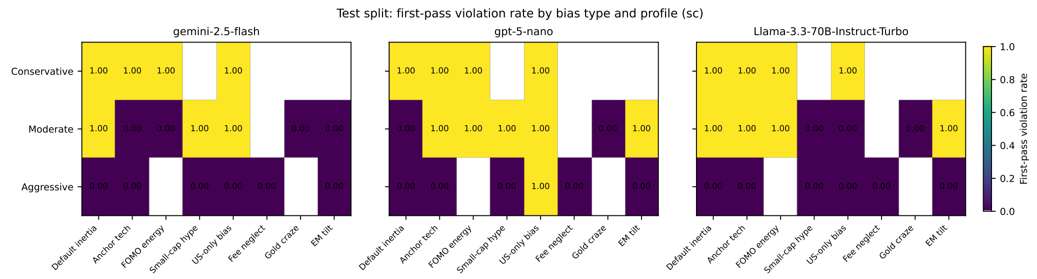

Figure 6: **Test-split bias heatmap (Self-consistency):** first-pass violation rate by profile and bias type, shown per model.

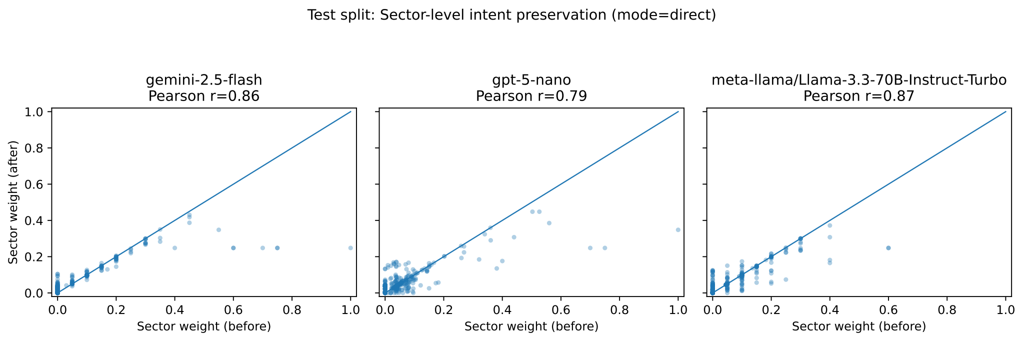

Figure 7: **Sector before/after (Direct, test split):** draft vs. projected sector allocations, shown per model (panels).

Interior margins and "polish" To reduce borderline numerical violations, we solve with small interior margins (e.g., $\tau = 2 \times 10^{-3}$ for asset/sector caps, and $hhi\_eps = 3 \times 10^{-4}$ for HHI). If the resulting solution is still slightly above the HHI cap due to rounding, we re-solve once with a tighter HHI bound and warm-start from the previous solution. As a final deterministic safeguard, we apply a tiny weight transfer (from the largest weight to a low-fee diversifier) if HHI remains marginally above the cap.

D.2 Solver parameter ablation and robustness

Table 8 varies numerical tolerances, iteration budgets. We report (i) solver success rate, (ii) final feasibility after projection, (iii) median correction distance $D = \|w^* - w_0\|_2$, and (iv) mean runtime. Across settings, the QCQP projection remains robust: SCS reliably solves the conic form induced by quadratic caps, while OSQP primarily serves as an optional fast-path when an instance reduces to a pure QP.

E Reproducibility + Parsing/Decoding

E.1 Implementation notes

We log model name, mode, prompt hash, seed, parse failures, and before/after metrics for reproducibility. We also run covariance and cap sanity checks (e.g., feasibility probes and eigenvalue diagnostics) during dataset creation; details are included in the appendix.

E.2 Decoding parameters

We use a strict JSON-only output contract for all modes. Temperatures are mode-specific but shared across models: direct T=0.10, critique T=0.20, and self-consistency T=0.28. When supported,

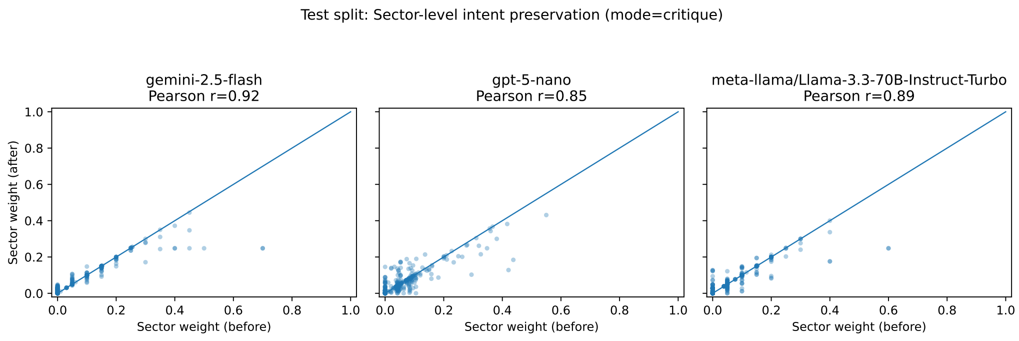

Figure 8: **Sector before/after (Critique, test split):** draft vs. projected sector allocations, shown per model.

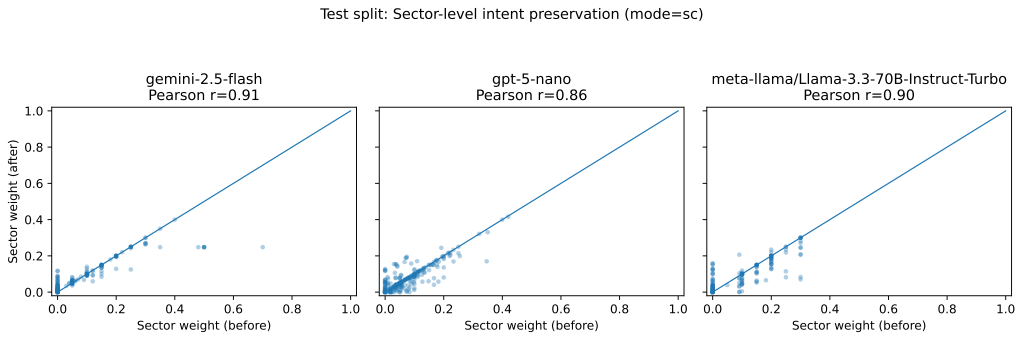

Figure 9: **Sector before/after (Self-consistency, test split):** draft vs. projected sector allocations, shown per model.

we set `top_p`=0.75 and `max_tokens`=1000. For GPT-5 nano, temperature/top-p are not exposed in the API used; we keep default sampling settings and set `reasoning_effort=low`.

E.3 Retry + parsing logic

Each query is attempted up to R=3 times if strict JSON parsing fails. A run is counted as a *parse failure* only if all R attempts fail. Parsing enforces a top-level object with exactly two keys: `weights` (a ticker-to-weight map) and `explanation` (a short string). Weights are coerced to nonnegative reals and renormalized to sum to 1. If the model returns a singleton list containing the object, we unwrap it.

F Statistical Details

For paired model comparisons on D, we use a two-sided Wilcoxon signed-rank test on per-scenario differences. Null hypothesis: H_0: the median paired difference in D between two models is 0. Alternative: H_1: the median paired difference is non-zero. We report the test statistic, the (BH-FDR) adjusted p-value, and an effect size r (rank-biserial / standardized) to emphasize *practical* as well as statistical significance.

Multiple testing. With three models, there are three model pairs per mode. Across three modes and three splits, this yields $3 \times 3 \times 3 = 27$ pairwise tests (and 36 including an “all-splits” aggregate view). We control false discovery using Benjamini–Hochberg FDR at $q = 0.10$ over the family of pairwise tests and report p_{FDR}.

Table 7: Paired Wilcoxon tests on D for all model pairs, reported by split and reasoning mode.

Split	Mode	Model A	Model B	N	r	p_{FDR}
train	critique	`gpt-5-nano`	`meta-llama/Llama-3.3-70B`	36	0.52	0.0054
train	critique	`gemini-2.5-flash`	`gpt-5-nano`	36	0.30	0.0707
train	critique	`gemini-2.5-flash`	`meta-llama/Llama-3.3-70B`	36	0.43	0.0162
train	direct	`gpt-5-nano`	`meta-llama/Llama-3.3-70B`	36	0.47	0.0122
train	direct	`gemini-2.5-flash`	`gpt-5-nano`	36	0.23	0.1686
train	direct	`gemini-2.5-flash`	`meta-llama/Llama-3.3-70B`	36	0.35	0.0581
train	sc	`gpt-5-nano`	`meta-llama/Llama-3.3-70B`	36	0.21	0.2021
train	sc	`gemini-2.5-flash`	`gpt-5-nano`	36	0.27	0.1463
train	sc	`gemini-2.5-flash`	`meta-llama/Llama-3.3-70B`	36	0.29	0.1463
dev	critique	`gpt-5-nano`	`meta-llama/Llama-3.3-70B`	15	0.56	0.0417
dev	critique	`gemini-2.5-flash`	`gpt-5-nano`	15	0.39	0.1186
dev	critique	`gemini-2.5-flash`	`meta-llama/Llama-3.3-70B`	15	0.58	0.0417
dev	direct	`gpt-5-nano`	`meta-llama/Llama-3.3-70B`	15	0.68	0.0352
dev	direct	`gemini-2.5-flash`	`gpt-5-nano`	15	0.36	0.1362
dev	direct	`gemini-2.5-flash`	`meta-llama/Llama-3.3-70B`	15	0.54	0.0512
dev	sc	`gpt-5-nano`	`meta-llama/Llama-3.3-70B`	15	0.33	0.1987
dev	sc	`gemini-2.5-flash`	`gpt-5-nano`	15	0.46	0.1137
dev	sc	`gemini-2.5-flash`	`meta-llama/Llama-3.3-70B`	15	0.44	0.1314
test	critique	`gpt-5-nano`	`meta-llama/Llama-3.3-70B`	21	0.62	0.0098
test	critique	`gemini-2.5-flash`	`gpt-5-nano`	21	0.36	0.1168
test	critique	`gemini-2.5-flash`	`meta-llama/Llama-3.3-70B`	21	0.61	0.0098
test	direct	`gpt-5-nano`	`meta-llama/Llama-3.3-70B`	21	0.54	0.0292
test	direct	`gemini-2.5-flash`	`gpt-5-nano`	21	0.21	0.3491
test	direct	`gemini-2.5-flash`	`meta-llama/Llama-3.3-70B`	21	0.42	0.0669
test	sc	`gpt-5-nano`	`meta-llama/Llama-3.3-70B`	21	0.33	0.2055
test	sc	`gemini-2.5-flash`	`gpt-5-nano`	21	0.28	0.2055
test	sc	`gemini-2.5-flash`	`meta-llama/Llama-3.3-70B`	21	0.30	0.2055

Table 8: Solver robustness ablation across different settings.

Setting	Solve success %	Final feasible %	Solved by OSQP %	Median D	Mean time (ms)	N
Default (paper)	100.000000	100.000000	0.000000	0.066900	61.320000	189
SCS tighter tol	100.000000	100.000000	0.000000	0.066900	59.290000	189
SCS looser tol	100.000000	100.000000	0.000000	0.066900	59.730000	189
More jitter	100.000000	100.000000	0.000000	0.066900	61.180000	189
No OSQP fast-path	100.000000	100.000000	0.000000	0.066900	58.160000	189

G ADDITIONAL DISCUSSION: UNIVERSE DESIGN AND GENERALIZATION

Universe design. We adopt a fixed 16-ETF universe as an *intentional design choice* for controlled diagnosis. It is large enough to express meaningful diversification patterns (U.S., ex-U.S., EM, sectors, bonds, gold) while remaining small enough to (i) make parsing failures and constraint-violation patterns interpretable, (ii) keep covariance and sector mappings stable and auditable, and (iii) enable systematic variation across bias prompts and risk profiles without confounding effects from universe size.

Generalization. The projection-based guardrail is not tied to 16 assets; it extends to larger universes given fees, sector mappings, and a covariance estimate. Generalization to hundreds of ETFs and time-varying risk models is left to future work, where scaling behavior, universe-dependent concentration effects, and covariance estimation error become central.