Towards Automated Causal Effect Estimation with Self-Evolving AI

Can Wang
Department of Biostatistics
Johns Hopkins Bloomberg School of Public Health
cwang271@jh.edu

Hongyu Zhao
Department of Computer Science
Johns Hopkins University
hzhao59@jhu.edu

Yiqun Chen
Departments of Biostatistics and Computer Science
Johns Hopkins University
yiqunc@jhu.edu

Abstract

Causal inference is central to scientific discovery, yet choosing appropriate methods remains challenging due to the complexity of statistical methodology and real-world data. Inspired by the success of artificial intelligence in accelerating scientific software, we introduce `InferenceEvolve`, an evolutionary framework that uses large language models (LLMs) to discover and iteratively refine causal estimators. Across four semi-synthetic causal benchmarks, `InferenceEvolve` improves over strong package baselines and automated causal systems under unbiased distributional summaries, nested search-split selection, and unseen out-of-distribution simulations, while achieving Pareto-frontier performance relative to 58 ACIC 2022 human submissions. We further study proxy-guided search when true effects are unavailable, finding strong proxy alignment across 243 discovered programs alongside informative benchmark-specific failure modes. Analysis of evolutionary trajectories shows that `InferenceEvolve` performs knowledge-guided discovery by recombining and refining established causal-estimation components in benchmark-specific ways. Our findings suggest that LLM-guided evolution could support methodological search when rewards are only partly observed, as in causal inference.

1 Introduction

Causal inference answers whether and how an intervention affects an outcome and has been central to scientific discovery and evidence-based decision-making (Hernán and Robins, 2020; Pearl, 2009; Bailey et al., 2024). One of the fundamental challenges in causal inference is the *counterfactual problem*: for any unit receiving treatment, we cannot observe what would have happened under control, and vice versa (Imbens and Rubin, 2015). Practitioners must therefore rely on assumptions and statistical methods to estimate treatment effects from observational data, where treatment assignment may depend on observed and unobserved confounders (Rosenbaum and Rubin, 1983). Despite decades of methodological progress—from propensity score methods (Stuart, 2010; Austin, 2011) to doubly robust estimators (Chernozhukov et al., 2018; Kennedy, 2024; Bang and Robins, 2005) and causal machine learning (Feuerriegel et al., 2024)—two persistent challenges remain. First, *which estimator to use*: selecting the right approach requires substantial expertise about both the statistical estimators and the substantive features of the data. Second, *how to evaluate estimators*. Since ground-truth treatment effects are unobservable in real data, researchers routinely rely on

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

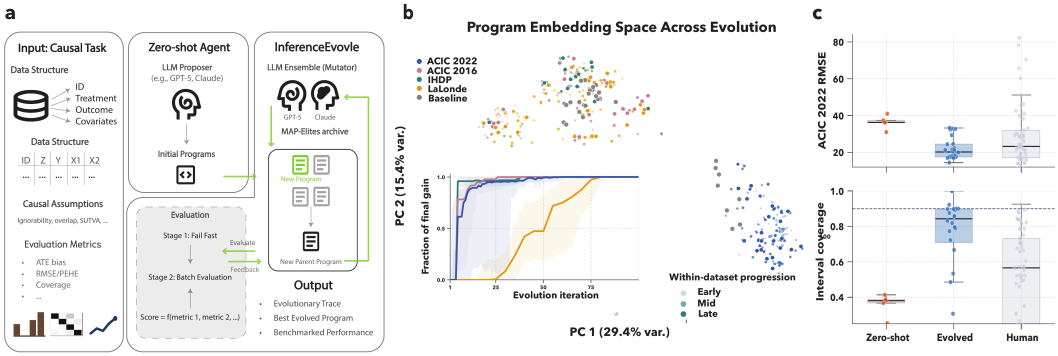


Figure 1: **Overview of `InferenceEvolve` for causal estimator discovery.** **a**, We start with a zero-shot program, which is iteratively improved by an LLM ensemble using benchmark-based feedback. **b**, Principal-component view of programs across evolutions (Colors = benchmarks and darker points = later checkpoints). The line graph summarizes best-so-far progress. **c**, Evolved programs compare favorably with human submissions on ACIC 2022 (Thal and Finucane, 2023). Top, RMSE distributions (lower better); bottom, empirical 90% interval coverage.

semi-synthetic benchmarks with known data-generating processes to assess estimators (Hill, 2011; Dorie et al., 2019; Thal and Finucane, 2023).

For a concrete example, consider estimating the effect of a job training program on subsequent earnings. Each individual i has potential outcomes $Y_i(1)$ and $Y_i(0)$ under treatment and control, but only one is observed. Stakeholders are often interested in the average treatment effect $\text{ATE} = \mathbb{E}[Y(1) - Y(0)]$, which requires estimating the missing potential outcomes. A human expert would typically proceed by: identifying the causal estimand (i.e., ATE in this case) under specific assumptions; choosing an estimation strategy (e.g., inverse propensity weighting, outcome regression, or doubly robust methods); selecting the models used to estimate nuisance parameters; and constructing confidence intervals. Each step involves modeling choices that can substantially affect the final estimate. To evaluate the quality of this pipeline, researchers often use semi-synthetic datasets that mimic the real-world settings they care about, and compute quantitative metrics such as bias, efficiency, and calibration of uncertainty (Imbens and Rubin, 2015).

The causal inference workflow is well-suited to automation for two reasons. First, the space of causal estimation strategies is combinatorially large but structured—different choices of identification strategy, nuisance models, and aggregation schemes can be composed programmatically. Second, the standard evaluation paradigm in causal inference already relies on simulated datasets with known ground truth, providing a natural fitness signal for optimization. Recent work has demonstrated the potential of large language model (LLM)-based agents for scientific tasks (Gao et al., 2024; Gottweis et al., 2025; Yamada et al., 2025; Swanson et al., 2025). In parallel, evolutionary code generation frameworks such as Autoresearch (Karpathy, 2026), AlphaEvolve (Novikov et al., 2025) and OpenEvolve (Sharma, 2025) have shown that LLMs can discover state-of-the-art algorithms through iterative program improvement guided by automated evaluation.

Related work: Recent LLM-based and automated work in causal inference has primarily evaluated or automated pieces of the causal-analysis workflow rather than searching over new treatment-effect estimators. CauSciBench evaluates whether LLMs can identify variables, select and implement an appropriate causal method, and report an effect estimate; Causal AI Scientist (CAIS), CATE-B, OpportunityFinder, CausalTune, and AutoCATE automate causal analysis by selecting adjustment sets, methods, estimators, or CATE pipelines from predefined workflow/search spaces; and NATURAL uses LLMs to extract or impute variables from unstructured text so that classical estimators can be applied (Acharya et al., 2025; Verma et al., 2025; Berrevoets et al., 2025; Nguyen et al., 2023; Kraev et al., 2022; Vanderschueren et al., 2025; Dhawan et al., 2024). AutoCATE is closest in spirit because it extensively validates doubly robust proxy scores for selecting among predefined CATE pipelines (preprocessing, 10 meta-learners, 9 base learners, hyperparameters, and ensembles—2,187 structures before hyperparameters); our setting instead searches executable estimator programs and compares directly against CAIS, AutoCATE, package baselines, and human submissions. Adjacent

work also uses LLM-guided evolution for causal structure learning, but the object being discovered is a graph score function rather than a treatment-effect estimator (Wang et al., 2025).

Contribution: Motivated by this gap, we introduce `InferenceEvolve`, an evolutionary framework that represents treatment-effect estimators as executable programs, evaluates them on benchmark tasks, and iteratively refines them through LLM-guided mutations (Figure 1). We note at the outset that `InferenceEvolve` performs *knowledge-guided discovery* of practically useful estimation strategies—combinatorially organizing established causal-inference components—rather than generating new causal identification theory. First, we define an open-ended estimator-discovery setting across ACIC 2022 (Thal and Finucane, 2023), ACIC 2016 (Dorie et al., 2019), IHDP (Hill, 2011), and LaLonde (LaLonde, 1986; Dehejia and Wahba, 1999), spanning panel, cross-sectional, semi-synthetic, and observational designs. Because these benchmarks provide many competition or semi-synthetic replicates, each run optimizes on one subset and reports only on disjoint held-out replicates (plus 50 newly generated out-of-distribution ACIC 2016 datasets; Supplementary Section S3.6). Second, we develop proxy guidance modes based on doubly robust pseudo-outcomes, enabling iterative refinement without ground-truth counterfactuals and validating proxy alignment across 243 discovered programs. Third, we analyze the resulting estimators, interpretability, and search budgets across model, split, and λ settings.

2 Methods

2.1 Benchmarks and evaluation metrics

We evaluated `InferenceEvolve` on four benchmark families that expose many semi-synthetic or competition replicates rather than a single fixed train/test table: ACIC 2022 Track 2 panel data (Thal and Finucane, 2023), ACIC 2016 (Dorie et al., 2019), IHDP (Hill, 2011), and semi-synthetic LaLonde/Dehejia–Wahba samples (LaLonde, 1986; Dehejia and Wahba, 1999). The LaLonde benchmark uses observed covariates and treatment assignment from the NSW/PSID job-training setting but evaluates against generated potential-outcome columns included in the released sample files and hidden from candidate programs. For the cross-sectional benchmarks, each candidate returns individual treatment-effect estimates $\hat{\tau}_i$ and an ATE estimate, and we report held-out precision in estimation of heterogeneous effects

$$\sqrt{\mathrm{PEHE}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{\tau}_i - \tau_i)^2}, \tag{1}$$

together with absolute ATE error. For ACIC 2022, whose target is a replicate-level patient-weighted sample average treatment effect on the treated, we report held-out RMSE across replicates, $\mathrm{RMSE} = \sqrt{R^{-1}\sum_{r=1}^{R}(\hat{\tau}_r - \tau_r)^2}$, and empirical 90% interval coverage. All main-text claims use the prespecified 20:80 split between search replicates ($\mathcal{D}_{\text{train}}$) and untouched held-out replicates ($\mathcal{D}_{\text{val}}$); matched 50:50 reruns and DGP split details (plus the 50-dataset ACIC 2016 outer test) are in Supplementary Sections S3.1 and S3.6.

2.2 `InferenceEvolve` search

`InferenceEvolve` treats an estimator as an executable Python program with a benchmark-specific `estimate(...)` interface. It builds on OpenEvolve (Sharma, 2025), using a four-island MAP-Elites archive (Mouret and Clune, 2015) (indexed by code length and lexical diversity) to retain high-scoring and diverse programs while an LLM ensemble proposes full-program mutations starting from a zero-shot seed P_0. Each evolutionary iteration (Algorithm 1) proceeds through six steps: (1) *Select parent and examples*, sampling a parent program P_{parent} and inspiration set $\mathcal{I}$ from the island archive $\mathcal{A}$; (2) *Compose prompt*, where ComposePrompt assembles the causal task specification, allowed columns and libraries (`pandas`, `numpy`, `scikit-learn`, `statsmodels`), required function signature, P_{parent} with its current score, a six-row data preview, top-3 archived programs, and $\mathcal{I}$ (Supplementary Section S1.2); (3) *Request full-program rewrite*, sampling an LLM ℓ from the two-model ensemble (primary GPT-5 or Gemini 3 Flash Preview with weight $w_1 = 0.80$; secondary Claude Sonnet 4 with $w_2 = 0.20$) to generate a complete rewritten candidate P'; (4) *Stage-1 quality-control screen*, executing P' on $\sim$10% of training replicates ($\mathcal{D}^{(1)}_{\text{train}}$) to filter crashes, timeouts, and

Algorithm 1 `InferenceEvolve` workflow for causal estimator discovery

Require: Seed program P_0, evaluator E, configuration C (with fail-fast threshold $C.\theta_1 = 0.001$), benchmark replicates $\mathcal{D}$
Ensure: Best program $P^\star$, evolution traces

Seed all RNGs with global seed; split $\mathcal{D}$ into training $\mathcal{D}_{\text{train}}$ and held-out $\mathcal{D}_{\text{val}}$.
Initialize the MAP-Elites archive $\mathcal{A}$ with $(P_0, E(P_0, \mathcal{D}_{\text{train}}))$.
Build LLM ensemble: $\{(w_1, \text{primary model}), (w_2, \text{secondary model})\}$.
for $i = 1 \ldots C.\text{max_iterations}$ **do**
  $(P_{\text{parent}}, \mathcal{I}) \leftarrow$ sample parent and inspiration set from $\mathcal{A}$.
  $q \leftarrow \textsc{ComposePrompt}(P_{\text{parent}}, \mathcal{I}, \text{metrics}, \text{top-}k \text{ programs})$.
  Sample LLM ℓ from ensemble with probabilities (w_1, w_2).
  $P' \leftarrow \ell(q)$ {Generate candidate program}
  Stage 1 (fail-fast): $m_1 \leftarrow E(P', \mathcal{D}^{(1)}_{\text{train}})$ on $\sim$10% of training replicates; $s_1 \leftarrow m_1[\texttt{combined_score}]$.
  if $s_1 < C.\theta_1$ **then**
    Discard P'; **continue**.
  end if
  Stage 2 (full): evaluate $E(P', \mathcal{D}_{\text{train}})$ on all training replicates.
  Compute composite score $S(P')$ using the dataset-specific scoring function.
  Insert $(P', S, \text{metrics})$ into $\mathcal{A}$; update MAP-Elites bins and $P^\star$.
end for
Final evaluation: run $E(P^\star, \mathcal{D}_{\text{val}})$ on held-out replicates.
return $P^\star$, evolution traces, evaluation logs.

degenerate outputs below a fixed fail-fast threshold $C.\theta_1 = 0.001$; (5) *Stage-2 full evaluation*, scoring surviving candidates on 100% of $\mathcal{D}_{\text{train}}$; and (6) *Retain useful versions and freeze*, updating the archive $\mathcal{A}$ and incumbent $P^\star$, then evaluating the frozen $P^\star$ once on $\mathcal{D}_{\text{val}}$ after 100 iterations.

For search-time selection, ACIC 2022 uses $S_{\text{ACIC2022}} = [1 + \log(1 + \text{RMSE}_{\mathcal{D}}) + \lambda \log(1 + |\text{Coverage}_{\mathcal{D}} - 0.9|/0.9)]^{-1}$, where $\text{RMSE}_{\mathcal{D}}$ and $\text{Coverage}_{\mathcal{D}}$ are aggregate metrics over the evaluated replicates. IHDP, ACIC 2016, and LaLonde use $S_{\text{ITE}} = [1 + \log(1 + \overline{\sqrt{\text{PEHE}}}) + \lambda \log(1 + \overline{|\text{ATE error}|})]^{-1}$, with overlines denoting averages over search replicates and $\lambda \in \{0.2, 1.0, 5.0\}$ (default $\lambda = 1.0$). Scores computed on the training data are used to guide evolution; all reported comparisons use *unseen held-out data*.

2.3 Proxy objectives, baselines, and program analysis

To test search without hidden counterfactuals, we also ran proxy-guided evolution. For IHDP, ACIC 2016, and LaLonde, the proxy objective combines 2-fold cross-fitted doubly robust pseudo-outcome fit, ATE discrepancy from a cross-fitted AIPW estimate, and normalized R-loss (Kennedy, 2023; 2024; Mahajan et al., 2024). For ACIC 2022, the proxy uses a doubly robust difference-in-differences pseudo-target with point-error, coverage, and interval-width terms. Full proxy formulas and implementation controls are given in Supplementary Section S1.

We compared against representative causal estimators and automated systems evaluated on the same held-out replicates with default parameters and no post-hoc selection: ordinary least squares / backdoor adjustment, inverse propensity weighting (Rosenbaum and Rubin, 1983; Austin, 2011), causal forests (Battocchi et al., 2019; Wager and Athey, 2018; Athey et al., 2019), BART (Hill, 2011), ACIC 2022 panel baselines, Causal AI Scientist (CAIS) (Verma et al., 2025), AutoCATE (Vanderschueren et al., 2025), and the 58 ACIC 2022 human competition submissions. The factorial sweep crossed four datasets, two primary LLMs (GPT-5 and Gemini 3 Flash Preview; Achiam et al., 2023; Team et al., 2024), three λ values, two held-out split regimes, and two replicate seeds, yielding 24 true-score final programs per dataset; matched proxy-guided runs used the same setup.

Finally, we analyzed final programs by source-code length, LLM-translated methods length, TF-IDF and embedding similarity to seven commit-pinned official reference wrappers, and a three-model LLM-as-judge taxonomy reviewed by two graduate students with causal-inference expertise. Details of prompts, allowed libraries, hyperparameters, reference wrappers, and judge aggregation appear in Supplementary Sections S1–S2.2.

3 Results

We first compared the evolved estimators against quantitative baselines, including the zero-shot LLM baseline, off-the-shelf causal inference methods, automated causal systems (Causal AI Scientist and AutoCATE), and, for ACIC 2022, the distribution of 58 human competition submissions. We

then qualitatively examined the learned programs, focusing on the estimator families that emerged at convergence, their overlap with existing work, and their coding and conceptual complexity.

3.1 `InferenceEvolve` finds strong candidate estimators

For the full true-score sweep, each dataset has 24 evolved final programs from a $2\times3\times2\times2$ factorial design over two primary LLMs from the GPT and Gemini families (GPT-5 and Gemini 3 Flash Preview; Achiam et al., 2023; Team et al., 2024), regularization weight $\lambda \in \{0.2, 1.0, 5.0\}$ (larger λ puts more weight on ATE estimation error and coverage error), held-out split (20% versus 50% training), and replicate seed. All main-text figures and tables focus on the prespecified 20:80 split; the matched 50:50 reruns are used only as sensitivity analyses in Supplementary Section S3.1. Search received feedback only from the training replicates, whereas every quantitative claim in the main text is evaluated on the untouched held-out replicates using unbiased distributional summaries (full distributions in Fig. 2; all-run means $\pm$ SE with default parameters and no post-hoc selection in Table 2), alongside nested search-split selection.

On the ACIC 2022 data with rich human competition results ($n = 58$), the 12 final estimators from different configurations of `InferenceEvolve` obtained a median root mean squared error (RMSE) 18.8 and median empirical coverage 0.84 (target level, 0.9), compared with 23.2 and 0.57, respectively, across the human submissions (Fig. 1). By RMSE, 8 of the 12 evolved final estimators outperformed the median human submission, nested search-split selection achieved RMSE 17.52 and coverage 0.89, and the top two evolved programs lie on the *Pareto frontier* with respect to RMSE and absolute deviation from nominal coverage (Miettinen, 1999). Without self-evolving improvement, zero-shot LLM performance was clearly weaker: GPT-5 zero-shot programs produced a difference-in-differences estimator with RMSE 33.1 and coverage 0.40, and the six zero-shot LLM seeds in the "Zero-shot LLM" box of Fig. 2d had median RMSE 36.2 (distinct from the 23.2 human median in Fig. 1c and 23.40 CAIS backdoor adjustment in Table 2; Supplementary Table 2).

Across all four benchmarks, Table 2 compares the unselected mean (SE) of `InferenceEvolve` under true-score and proxy guidance against the strongest package or Causal AI Scientist (CAIS) comparator and **AutoCATE** (Vanderschueren et al., 2025), evaluated on the identical held-out replicates using default parameters with no post-hoc selection. Even without selection, `InferenceEvolve` beats the strongest package/CAIS baseline on every ITE metric in the mean and median, and nested selection (choosing a single configuration strictly by its training-split score S) reduces error by 22–43% across all four benchmarks. Compared with AutoCATE, `InferenceEvolve` is competitive on $\sqrt{\text{PEHE}}$ (1.399 vs. 1.402 on IHDP; 0.706 vs. 0.760 on LaLonde; 0.963 vs. 1.445 on ACIC 2016) and significantly reduces ATE estimation error across all benchmarks (0.086/0.084 vs. 0.215 on IHDP; 0.118 vs. 0.199 on ACIC 2016; 0.057 vs. 0.095 on LaLonde), since `InferenceEvolve` jointly optimizes ATE and PEHE (or SATT RMSE and coverage) over executable programs whereas AutoCATE tunes a CATE pipeline for PEHE alone and cannot natively handle panel data like ACIC 2022 or output uncertainty intervals.

To further test generalization beyond the fixed benchmark splits without any post-hoc selection, we froze the final ACIC 2016 `InferenceEvolve` program after evolution finished and evaluated it once on 50 newly generated ACIC 2016 datasets balanced across 5 unseen DGP settings (10 fresh simulations per setting; Table 1). Pooled across all 50 new datasets, the frozen program achieved $\sqrt{\text{PEHE}}$ 0.986 ± 0.125 and ATE error 0.121 ± 0.023, outperforming Causal Forest (1.428 ± 0.190 and 0.219 ± 0.041) and achieving lower ATE error in 5 of 5 settings and lower $\sqrt{\text{PEHE}}$ in 3 of 5 settings. Notably, the outer-test $\sqrt{\text{PEHE}}$ (0.986) closely tracks the unselected all-run mean (0.963) and nested search-split score (0.919) while sitting above the exploratory post-hoc best-of-12 score (0.858), confirming that removing post-hoc selection eliminates winner's-curse optimism while preserving `InferenceEvolve`'s advantage on unseen DGP realizations.

The final programs produced by `InferenceEvolve` tended to be longer in code length and more complex in terms of methodology than the off-the-shelf baselines. This increase in complexity should be interpreted together with the evaluation protocol: the search never saw the held-out metrics reported in Fig. 2 and Table 2; those numbers were computed only after the final program was frozen. When GPT-5 was used to transcribe the final code into manuscript-style methods paragraphs, the resulting descriptions were consistently shortest for the baseline programs and longer for the true-evolved and proxy-evolved programs across all four datasets; the same ordering was also evident

Table 1: **DGP-stratified outer test on 50 newly generated ACIC 2016 datasets.** A single frozen `InferenceEvolve` program is evaluated once, without retraining or selection, on 10 newly generated simulations from each of 5 unseen DGP settings. Values report mean (SE).

ACIC 2016 setting	`InferenceEvolve` $\sqrt{\text{PEHE}}$	Causal forest $\sqrt{\text{PEHE}}$	`InferenceEvolve` ATE error	Causal forest ATE error
All 50 new datasets (pooled)	**0.986 (0.125)**	1.428 (0.190)	**0.121 (0.023)**	0.219 (0.041)
Setting 1 ($n = 10$)	**0.765 (0.123)**	1.452 (0.201)	**0.149 (0.065)**	0.287 (0.116)
Setting 2 ($n = 10$)	0.808 (0.271)	**0.396 (0.063)**	**0.045 (0.008)**	0.091 (0.022)
Setting 3 ($n = 10$)	0.394 (0.023)	**0.286 (0.047)**	**0.086 (0.015)**	0.112 (0.026)
Setting 4 ($n = 10$)	**1.517 (0.352)**	2.605 (0.494)	**0.061 (0.014)**	0.106 (0.019)
Setting 5 ($n = 10$)	**1.447 (0.328)**	2.401 (0.415)	**0.265 (0.081)**	0.500 (0.129)

Table 2: **Unbiased held-out comparison against package/CAIS baselines and AutoCATE, plus nested selection.** All systems are evaluated on identical held-out replicates with true metrics. The first four numeric columns report **mean (SE)** over all runs using default parameters with **no selection**; *IE nested* selects a single configuration strictly by training-split score S. ‡On ACIC 2022, AutoCATE requires pre/post panel collapsing and outputs no intervals.

Benchmark	Metric	Best baseline / CAIS	AutoCATE	IE — true-score	IE — proxy	IE nested (vs. base.)
IHDP	$\sqrt{\text{PEHE}}$	2.434 (0.650) BART	1.402 (0.330)	**1.399 (0.035)**	1.632 (0.311)	1.366 (-43%)
IHDP	ATE error	0.111 (0.018) BART	0.215 (0.069)	0.086 (0.006)	**0.084 (0.003)**	0.072 (-35%)
ACIC 2016	$\sqrt{\text{PEHE}}$	1.321 (0.085) Causal forest	1.445 (0.092)	**0.963 (0.032)**	2.416 (0.389)	0.919 (-28%)
ACIC 2016	ATE error	0.148 (0.015) Causal forest	0.199 (0.019)	**0.118 (0.010)**	0.172 (0.012)	0.089 (-40%)
LaLonde	norm. $\sqrt{\text{PEHE}}$	0.769 (0.007) BART	0.760 (0.017)	**0.706 (0.019)**	0.754 (0.017)	0.598 (-22%)
LaLonde	norm. ATE error	0.073 (0.011) BART	0.095 (0.024)	**0.057 (0.005)**	0.070 (0.005)	0.033 (-31% vs. CAIS)
ACIC 2022	RMSE (overall SATT)	23.40 (2.29) CAIS backdoor	33.68 (1.59)‡	26.62 (2.46)	**19.72 (1.33)**	17.52 (-25%, cov. 0.89)

in code line counts (Fig. 2). This pattern was robust to the choice of transcription LLM, including Gemini 2.5 Pro and Claude Sonnet 4.5 (Supplementary Figs. 1 and 2).

3.2 Evolution discovers dataset-specific and interpretable estimator families

Beyond held-out performance, Figure 3 shows that `InferenceEvolve` performs *knowledge-guided discovery*: 1. different benchmarks converge to recognizable estimator families; and 2. gains come from combinatorially organizing components from established causal-inference methods rather than inventing new causal theory.

We first note that the converged pool of estimators were strongly dataset-specific, which explains the distinct clusters in PCA plot in Figure 1: since it is a panel data with the same units across timepoints, ACIC 2022 concentrated almost entirely on OLS / panel estimators (21/24). By contrast, ACIC 2016 converged on X-learners (Künzel et al., 2019) (18/24), IHDP on X- and T-learners (Künzel et al., 2019) (15/24 and 6/24), and LaLonde on DR-learner (Kennedy, 2023) and ensemble hybrids (13/24 and 5/24). Across all 96 final programs, the majority-vote family counts were X-learner (34), OLS (21), DR-learner (19), T-learner (11), and other (11). The three judges in the LLM-as-a-judge pipeline agreed on the coarse family label for 84/96 programs, but agreed on a single nearest named published method for only 42/96, indicating convergence at the level of method families and components rather than exact textbook replication.

Closest official-reference TF-IDF cosine increased from an overall median of 0.446 among the zero-shot baseline programs to 0.640 by checkpoint 20, then remained nearly flat through checkpoint 100 (median 0.638). This pattern is consistent with rapid movement toward familiar estimator families followed by benchmark-specific refinement. Relative to the closest official references, the most common refinements were cross-fitting or out-of-fold nuisance estimation (41/96 programs), robust uncertainty or coverage calibration (32/96), patient- or practice-weighted aggregation (19/96), estimator ensembling or hybridization (17/96), constant or proxy propensity models (15/96), and propensity clipping or overlap control (14/96)—for example, combining random-forest propensity weighting of X-learner arms, a 60/40 X-/R-learner CATE blend shifted to match an AIPW ATE target, and Mundlak-style practice-mean controls (Table 3). Pairwise code similarity among evolved programs again showed dataset-specific convergence: overall TF-IDF similarity was

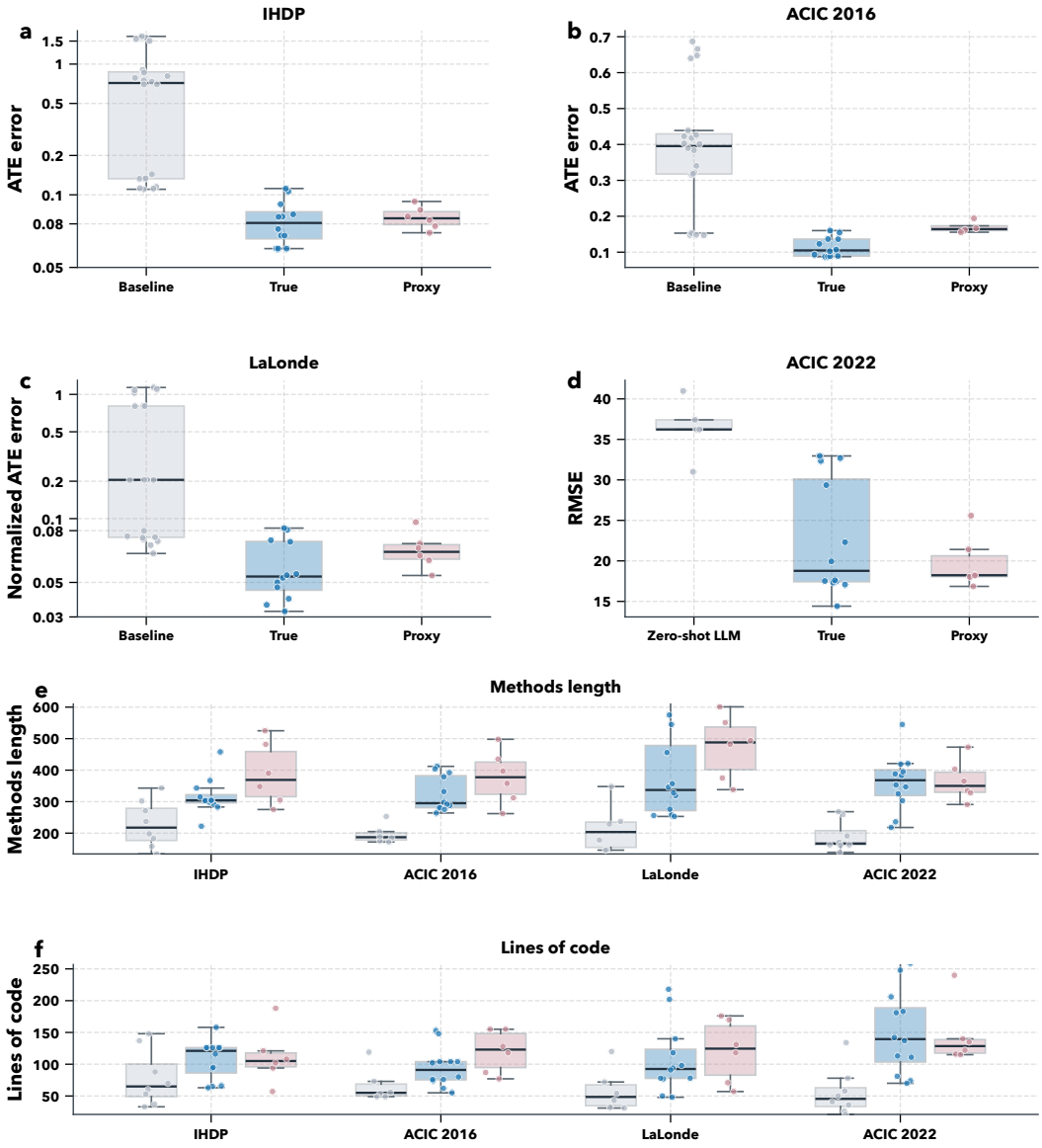


Figure 2: **Held-out performance distributions for baselines, true-evolved programs, and proxy-evolved programs, with text- and code-complexity summaries.** Panels **a**–**d** show the metric distributions on the held-out set for baseline methods and the final true-evolved and proxy-evolved runs (in **d**, "Zero-shot LLM" shows zero-shot LLM seeds with median RMSE 36.2). Panels **e** and **f** summarize GPT-5 methods length and code lines by dataset.

higher within datasets than between datasets (0.359 versus 0.228), and the same ordering held for `text-embedding-3-large` embeddings (0.776 versus 0.665).

These structural shifts were accompanied by larger programs and longer scientific write-ups, while remaining interpretable at both the family level and the step-by-step trajectory level (Supplementary Sections S4–S5). Across datasets, baseline programs had a median size of 54 lines and 2,004 characters, compared with 104 lines and 3,922 characters for true-evolved programs and 119.5 lines and 4,698.5 characters for proxy-evolved programs (Supplementary Fig. 1). Translating each program into a manuscript-style methods paragraph with GPT-5, Gemini 2.5 Pro, or Claude Sonnet 4.5 yielded the same ordering: median description length increased from 172.5 words for baselines to 263.0 for true-evolved programs and 279.5 for proxy-evolved programs (Fig. 2). Table 3 summarizes representative solution families relative to the closest official reference wrapper.

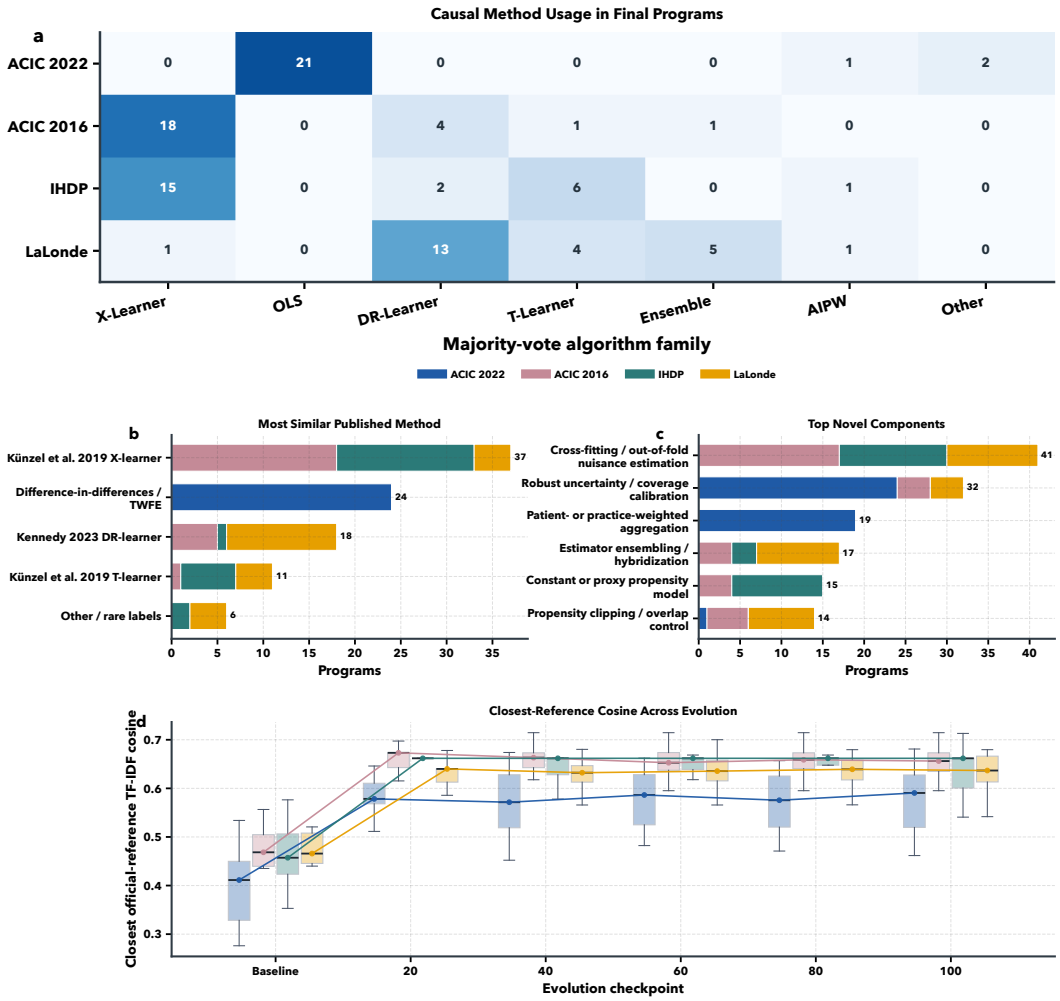


Figure 3: `InferenceEvolve` **converges to dataset-specific estimator families through knowledge-guided recombination. a**, Majority-vote algorithm-family assignments for the 96 final evolved programs. **b**, Majority-vote most similar published method. **c**, Novel algorithmic components extracted from the union of LLM-as-a-judge reports for each program. **d**, Tokenized TF-IDF cosine similarity between each program and its closest official reference wrapper across zero-shot baselines and checkpoints, with boxplots stratified by dataset.

Table 3: **Representative solution families discovered by evolution.** Final programs cluster into dataset-specific estimator families with recognizable refinement patterns.

Dataset	Dominant family	Typical refinements during evolution	Within-dataset TF-IDF	Closest official reference	Median closest-ref. emb. cosine
ACIC 2022	TWFE / DiD / panel FE	Pre-period anchoring, detrending, weighting, clustered inference, calibration	0.336 ± 0.110	IPW	0.585
ACIC 2016	Meta-learners + DR	Cross-fitting, clipping, covariate encoding, X-/DR learners, and AIPW aggregation	0.353 ± 0.173	DR-learner	0.678
IHDP	Boosted/forest T-/X-learners	Outcome tuning, DR pseudo-outcomes, ATE calibration, and stable nuisance fits	0.415 ± 0.244	Causal forest	0.691
LaLonde	Weighting/DR hybrids for observational bias	Propensity weighting, AIPW/DR correction, adaptive regularization, and small-sample ensembling	0.331 ± 0.116	DR-learner	0.707

Search dynamics were benchmark-dependent: on average, best-so-far scores reached 76% of their final gain by iteration 25 and 90% by iteration 50, with the median run reaching 90% of its final

value within 20 iterations on IHDP (iteration 5), ACIC 2016 (iteration 12), and ACIC 2022 (iteration 16), whereas LaLonde reached the same threshold around iteration 75 (Supplementary Fig. 6). Per 100-iteration run, search issued 100 generation calls consuming $\sim$1.2M input and 0.2M output tokens ($\sim$\$2–\$4 at list prices without caching, and $\sim$5.5 h of wall-clock time dominated by estimator evaluation); capping search at 20–50 iterations reduces this budget further. On ACIC 2022, the strongest run moved from an under-calibrated weighted TWFE seed to an augmented DiD estimator by adding pre-period anchoring, post-period differencing, trend adjustment, and cluster-robust inference, reducing RMSE from 33.1 at initialization to 15.2 at its final best checkpoint (Supplementary Sections S3.2 and S4).

3.3 Proxy-guided optimization

A central difficulty in causal estimator discovery is that the true reward is usually unobserved in real data. We therefore replaced hidden-truth objectives with observed-data proxy rewards built from cross-fitted doubly robust pseudo-outcomes for IHDP, ACIC 2016, and normalized LaLonde, and from a DR difference-in-differences pseudo-target for ACIC 2022 (see Methods). This lets the same evolutionary loop operate even when ground-truth counterfactuals are unavailable.

Evaluating all 49 proxy-guided runs across every unique best-so-far program discovered during search (243 programs total; Supplementary Table 7), proxy-to-true combined-score correlation was $r = 0.92$ on IHDP ($n = 55$), $r = 0.92$ on ACIC 2016 ($n = 69$), and $r = 0.89$ on normalized LaLonde ($n = 65$), while on ACIC 2022 ($n = 54$) the DR-DID hit-rate and point-error components achieved Spearman $\rho = 0.96$ with true coverage and $\rho = 0.86$ with true SATT RMSE. A proxy-design sensitivity analysis over all 243 programs showed that proxy–truth correlation remained stable across cross-fitting fold counts ($K \in \{2, 5, 10\}$), propensity clipping windows, and outcome nuisance learners, whereas replacing the doubly robust DR-DID pseudo-target on ACIC 2022 with naive DiD reduced correlation to 0.67 and 0.63 (Supplementary Tables 8–9). Even so, proxy-guided search beat package and AutoCATE baselines on IHDP, LaLonde $\sqrt{\text{PEHE}}$, and ACIC 2022 RMSE while underperforming on ACIC 2016 $\sqrt{\text{PEHE}}$ (Table 2), indicating that general observed-data proxy design remains an open problem.

4 Discussion, Conclusion, and Limitations

With the growing scientific and coding capabilities of language models, their use in causal inference offers a promising route for automating empirical causal analysis. We introduced `InferenceEvolve`, a knowledge- and feedback-guided evolutionary framework that searches causal-inference code spaces for empirically stronger estimators. Across four widely used semi-synthetic benchmarks, `InferenceEvolve` improved over fixed package baselines and automated causal systems (CAIS and AutoCATE) under unbiased all-run means, medians, nested search-split selection, and a 50-dataset out-of-distribution test on ACIC 2016, while achieving Pareto-frontier performance relative to 58 human competition submissions on ACIC 2022.

Comparing `InferenceEvolve` with automated systems such as AutoCATE and CAIS highlights the trade-off between open-ended program evolution and structured pipeline selection. AutoCATE selects and tunes a CATE pipeline over a predefined space (preprocessing, 10 meta-learners, 9 base learners, hyperparameters, and ensembles—2,187 pipeline structures before hyperparameters) optimized for PEHE via T-risk, whereas `InferenceEvolve` searches executable programs that can jointly modify nuisance learners, feature engineering, cross-fitting, weighting, ATE calibration, and interval construction. Searching over code increases search variance and compute cost ($\sim$1.2M input and 0.2M output tokens, or $\sim$\$2–\$4 in API charges and $\sim$5.5 h of evaluation per 100-iteration run), but provides two practical advantages: it accommodates arbitrary causal estimands and study designs—such as longitudinal panel data with uncertainty intervals on ACIC 2022, where AutoCATE cannot be applied natively—and it substantially reduces ATE estimation error by synthesizing explicit doubly robust calibration steps alongside CATE estimation. Moreover, because best-so-far scores reach 76% of their final gain by iteration 25 and 90% by iteration 50 on average (and within 5–20 iterations on three of four benchmarks), practitioners can cap search at 20–50 iterations to capture most of the benefit at a fraction of the budget.

Our qualitative and trajectory analyses also clarify what self-evolving LLMs discover in practice. Rather than inventing new causal identification theory, `InferenceEvolve` performs knowledge-

guided discovery by recombining established methodological families (X-learners, DR-learners, and panel estimators) with data-driven refinements that are auditable commit by commit. As a direction for future work, this two-level interpretability—mapping each estimator to a broad family while isolating individual code edits along its evolutionary trajectory—could be turned into a constructive testbed: isolating interpreted components that correlate with improved causal estimation and testing whether supplying those concepts directly to the LLM yields stronger estimators with less search.

Finally, our findings clarify the scope of generalization and the open challenge of proxy-guided search. Our held-out and 50-dataset outer-test evaluations establish generalization across independent samples from the same mechanism (Level 1) and across unseen DGP realizations within a benchmark family under fixed schema and causal assumptions (Level 2). They do not establish zero-shot transfer of a frozen code artifact to a new study design, covariate schema, or identification regime (Level 3), where `InferenceEvolve` is intended to be rerun on the new task specification. Moving to real-world observational studies without ground-truth counterfactuals (Level 4) relies on observed-data proxies: while our 243-program sensitivity analysis shows that doubly robust proxy–truth correlations are remarkably stable across cross-fitting folds, clipping windows, and nuisance learners, high candidate correlation does not prevent proxy overfitting on every benchmark (as seen on ACIC 2016 $\sqrt{\text{PEHE}}$). We therefore view `InferenceEvolve` as a flexible methodological search tool to complement—not replace—human domain expertise, study-design validation, and sensitivity analysis in high-stakes policy and health applications.

Acknowledgements

We would like to thank the Causal Inference Working Group in the Department of Biostatistics at Johns Hopkins University and Alex Luedtke for helpful feedback. This work was partially supported by a Data Science and AI Faculty Innovation Fund in the Department of Biostatistics at the Johns Hopkins Bloomberg School of Public Health and a Google Academic Research Award.

References

Sawal Acharya, Terry Jingchen Zhang, Andrew Kim, Anahita Haghighat, Xianlin Sun, Rahul Babu Shrestha, Maximilian Mordig, Furkan Danisman, Clijo Jose, Yahang Qi, Pepijn Cobben, Bernhard Schölkopf, Mrinmaya Sachan, and Zhijing Jin. CauSciBench: Assessing LLM Causal Reasoning for Scientific Research. In *NeurIPS 2025 Workshop on CauScien: Uncovering Causality in Science*, October 2025. URL `https://openreview.net/forum?id=EO8mTLqDuT`.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Susan Athey, Julie Tibshirani, and Stefan Wager. Generalized random forests. *The Annals of Statistics*, 47(2):1148–1178, 2019. doi: 10.1214/18-AOS1709.

Peter C Austin. An introduction to propensity score methods for reducing the effects of confounding in observational studies. *Multivariate Behavioral Research*, 46(3):399–424, 2011.

Drew H Bailey, Alexander J Jung, Adriene M Beltz, Markus I Eronen, Christian Gische, Ellen L Hamaker, Konrad P Kording, Catherine Lebel, Martin A Lindquist, Julia Moeller, et al. Causal inference on human behaviour. *Nature Human Behaviour*, 8(8):1448–1459, 2024.

Heejung Bang and James M Robins. Doubly robust estimation in missing data and causal inference models. *Biometrics*, 61(4):962–973, 2005.

Keith Battocchi, Eleanor Dillon, Maggie Hei, Greg Lewis, Paul Oka, Miruna Oprescu, and Vasilis Syrgkanis. EconML: A Python Package for ML-Based Heterogeneous Treatment Effects Estimation. GitHub repository, 2019. URL `https://github.com/py-why/EconML`.

Jeroen Berrevoets, Julianna Piskorz, Rob Davis, Harry Amad, Jim Weatherall, and Mihaela van der Schaar. Technical report: Facilitating the adoption of causal inference methods through LLM-empowered co-pilot. *arXiv preprint arXiv:2508.10581*, 2025. doi: 10.48550/arXiv.2508.10581. URL `https://arxiv.org/abs/2508.10581`.

Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. Double/debiased machine learning for treatment and structural parameters. *The Econometrics Journal*, 21(1):C1–C68, 2018. doi: 10.1111/ectj.12097.

Rajeev H Dehejia and Sadek Wahba. Causal effects in nonexperimental studies: Reevaluating the evaluation of training programs. *Journal of the American Statistical Association*, 94(448):1053–1062, 1999.

Nikita Dhawan, Leonardo Cotta, Karen Ullrich, Rahul G. Krishnan, and Chris J. Maddison. End-to-end causal effect estimation from unstructured natural language data. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://proceedings.neurips.cc/paper_files/paper/2024/hash/8d2aad53fa148791ffd004a7c9cce5ad-Abstract-Conference.html.

Vincent Dorie, Jennifer Hill, Uri Shalit, Marc Scott, and Dan Cervone. Automated versus do-it-yourself methods for causal inference: Lessons learned from a data analysis competition. *Statistical Science*, 34(1):43–68, 2019.

Stefan Feuerriegel, Dennis Frauen, Valentyn Melnychuk, Jonas Schweisthal, Konstantin Hess, Alicia Curth, Stefan Bauer, Niki Kilbertus, Isaac S Kohane, and Mihaela van der Schaar. Causal machine learning for predicting treatment outcomes. *Nature Medicine*, 30(4):958–968, 2024.

Shanghua Gao, Ada Fang, Yepeng Huang, Valentina Giunchiglia, Ayush Noori, Jonathan Richard Schwarz, Yasha Ektefaie, Jovana Kondic, and Marinka Zitnik. Empowering biomedical discovery with AI agents. *Cell*, 187(22):6125–6151, 2024.

J Gottweis, WH Weng, A Daryin, T Tu, A Palepu, P Sirkovic, and V Natarajan. Towards an AI co-scientist. *arXiv preprint arXiv:2502.18864*, 2025.

Miguel A Hernán and James M Robins. *Causal Inference: What If*. Chapman & Hall/CRC, 2020.

Jennifer L. Hill. Bayesian Nonparametric Modeling for Causal Inference. *Journal of Computational and Graphical Statistics*, 20(1):217–240, January 2011. ISSN 1061-8600. doi: 10.1198/jcgs.2010.08162. URL https://doi.org/10.1198/jcgs.2010.08162. Publisher: ASA Website _eprint: https://doi.org/10.1198/jcgs.2010.08162.

Guido W Imbens and Donald B Rubin. *Causal inference in statistics, social, and biomedical sciences*. Cambridge University Press, 2015.

Andrej Karpathy. autoresearch, 2026. URL https://github.com/karpathy/autoresearch. GitHub repository, accessed 2026-04-05.

Edward H Kennedy. Towards optimal doubly robust estimation of heterogeneous causal effects. *Electronic Journal of Statistics*, 17(2):3008–3049, 2023.

Edward H Kennedy. Semiparametric doubly robust targeted double machine learning: a review. *Handbook of Statistical Methods for Precision Medicine*, pages 207–236, 2024.

Egor Kraev, Timo Flesch, Hudson Taylor Lekunze, Mark Harley, and Pere Planell Morell. Out-of-sample scoring and automatic selection of causal estimators. *arXiv preprint arXiv:2212.10076*, 2022. doi: 10.48550/arXiv.2212.10076. URL https://arxiv.org/abs/2212.10076.

Sören R Künzel, Jasjeet S Sekhon, Peter J Bickel, and Bin Yu. Metalearners for estimating heterogeneous treatment effects using machine learning. *Proceedings of the National Academy of Sciences*, 116(10):4156–4165, 2019.

Robert J LaLonde. Evaluating the econometric evaluations of training programs with experimental data. *The American Economic Review*, 76(4):604–620, 1986.

Divyat Mahajan, Ioannis Mitliagkas, Brady Neal, and Vasilis Syrgkanis. Empirical analysis of model selection for heterogeneous causal effect estimation. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=yuy6cGt3KL.

Kaisa Miettinen. *Nonlinear multiobjective optimization*, volume 12. Springer Science & Business Media, 1999.

Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

Huy Nguyen, Prince Grover, and Devashish Khatwani. Opportunityfinder: A framework for automated causal inference, 2023. URL https://arxiv.org/abs/2309.13103.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Judea Pearl. Causal inference in statistics: An overview. *Statistics Surveys*, 3:96–146, 2009.

Paul R Rosenbaum and Donald B Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55, 1983.

Asankhaya Sharma. Openevolve: an open-source evolutionary coding agent. https://github.com/codelion/openevolve, 2025.

Elizabeth A Stuart. Matching methods for causal inference: A review and a look forward. *Statistical Science*, 25(1):1, 2010.

Kyle Swanson, Wesley Wu, Nash L Bulaong, John E Pak, and James Zou. The virtual lab of AI agents designs new sars-cov-2 nanobodies. *Nature*, 646(8085):716–723, 2025. doi: 10.1038/s41586-025-09442-9.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

Dan R.C. Thal and Mariel M. Finucane. Causal Methods Madness: Lessons Learned from the 2022 ACIC Competition to Estimate Health Policy Impacts. *Observational Studies*, 9(3):3–27, 2023. ISSN 2767-3324. URL https://muse.jhu.edu/pub/56/article/895650. Publisher: University of Pennsylvania Press.

Toon Vanderschueren, Tim Verdonck, Mihaela van der Schaar, and Wouter Verbeke. AutoCATE: End-to-end, automated treatment effect estimation. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 60880–60904. PMLR, 2025. URL https://proceedings.mlr.press/v267/vanderschueren25a.html.

Vishal Verma, Sawal Acharya, Devansh Bhardwaj, Samuel Simko, Yongjin Yang, Anahita Haghighat, Dominik Janzing, Mrinmaya Sachan, Bernhard Schölkopf, and Zhijing Jin. Causal AI scientist: Facilitating causal data science with large language models. In *NeurIPS 2025 Workshop on CauScien: Uncovering Causality in Science*, 2025. URL https://openreview.net/forum?id=EDWTHMVOCj.

Stefan Wager and Susan Athey. Estimation and inference of heterogeneous treatment effects using random forests. *Journal of the American Statistical Association*, 113(523):1228–1242, 2018.

Zidong Wang, Fei Liu, Qi Feng, Qingfu Zhang, and Xiaoguang Gao. LLM-enhanced score function evolution for causal structure learning. In James Kwok, editor, *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pages 9086–9094. International Joint Conferences on Artificial Intelligence Organization, 8 2025. doi: 10.24963/ijcai.2025/1010. URL https://doi.org/10.24963/ijcai.2025/1010. Main Track.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The AI scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025.

Data availability

The benchmark datasets used in this study are publicly available from their original sources: ACIC 2022 (`https://acic2022.mathematica.org/data-challenge/`), ACIC 2016 (`https://github.com/vdorie/aciccomp`), the IHDP benchmark replications (`https://github.com/AMLab-Amsterdam/CEVAE`), and the Dehejia–Wahba / LaLonde data distributed through CRAN packages such as `Matching` (`https://search.r-project.org/CRAN/refmans/Matching/html/lalonde.html`). Exact download instructions, preprocessing steps, and benchmark-specific wrappers are documented in the project repository. The release documentation will list source licenses and terms of use for each dataset and package, including Creative Commons Attribution 4.0 (CC BY 4.0) where indicated by benchmark-data providers and package licenses such as GPL-3 for CRAN assets. An interactive exploration of the benchmark results and evolved programs is available at `https://yiqunchen.github.io/causal-agent/`.

Code availability

Code for `InferenceEvolve`, including prompts, evaluation scripts, figure generation, and analysis workflows, is available at `https://github.com/yiqunchen/causal-agent`. The repository also documents dataset access, preprocessing, per-run configurations, asset licenses and terms of use, and the final evolved programs used in this manuscript. The de-anonymized code release is intended to use CC BY 4.0, or the closest compatible open license permitted by repository dependencies.

A Supplementary Material

This supplementary section is organized to parallel the main text. We first document the exact prompts, search workflow, and scoring rules that define the estimator-discovery problem. We then collect the additional benchmark summaries and sensitivity analyses referenced from the Results section, before turning to dataset-specific evolutionary trajectories and representative code edits.

A.1 S1. Search setup and prompts

This section defines the estimator-search problem solved by `InferenceEvolve`. We begin with the exact prompt templates supplied to the LLMs, then summarize the common workflow, the fixed configuration choices, the benchmark-specific scoring rules, and the implementation details needed to reproduce the experiment.

A.1.1 S1.1. Evolution prompts

The four prompt templates below were used to initialize the evolutionary search across benchmarks. Each prompt fixes the data schema, return signature, evaluation metric, allowed libraries, and mutation rules; a short data preview was appended dynamically at runtime.

ACIC 2022 Evolution Prompt

```
You are an expert causal inference statistician and Python programmer.

Background:
- We have multisite randomized trial data in long format (practice-year level).
- Treatment is assigned at the practice level, randomized, and 'post' marks the
  period after treatment rollout.

Data columns:
    - 'id.practice' (int): site identifier
    - 'year' (int): panel time index
    - 'Y' (float): average monthly spending per patient
    - 'Z' (0/1): treatment indicator
    - 'post' (0/1): post-treatment period indicator
    - 'n.patients' (int): patient count (weights)
    - Practice-level covariates: X1..X9
    - Aggregated patient covariates: V1_avg..V5_C_avg

Task: Estimate the Overall SATT for treated practices in the post-treatment
period, weighted by n.patients.

OpenEvolve Objective:
- Coverage gap is |coverage - 0.9|/0.9.
- Score = log(1+RMSE) + lambda * log(1+coverage_gap).
- Combined score reported to OpenEvolve is 1/(1+score).
- Minimise score by reducing RMSE while achieving close to 90% coverage.

Function signature:
    def estimate(df) -> (tau_hat, lb, ub, method)

Constraints: Pure, deterministic. Use only pandas, numpy, scikit-learn,
statsmodels. Robust to NaNs and edge cases. Modify only the EVOLVE-BLOCK.

[Data preview appended at runtime]
```

IHDP Evolution Prompt

```
You are an expert causal inference researcher and Python programmer working on
the IHDP semi-synthetic benchmark.

Dataset schema: ‘treatment‘ (binary), ‘y_factual‘ (outcome), ‘x1‘..‘x25‘
(covariates). Ground-truth columns removed before execution.

Task: Implement a deterministic estimator returning (ite_hat, ate_hat,
method_name).

Evaluation: Score = log(1+mean_sqrt_PEHE) + lambda*log(1+mean_ATE_err).
The combined score reported to OpenEvolve is 1/(1+score).
50 replicates; configurable train/test split.

Objective: Minimise score. Explore diverse estimator families (T-/X-/DR-learners,
doubly-robust, etc.). Prioritise stability.

Requirements: Pure function. pandas, numpy, scikit-learn, statsmodels only.
Modify only the EVOLVE-BLOCK.

[Data preview appended at runtime]
```

ACIC 2016 Evolution Prompt

```
You are an expert causal inference researcher and Python programmer working on
the ACIC 2016 semi-synthetic benchmark.

Dataset schema: ‘treatment‘ (binary), ‘y_factual‘ (outcome), ‘x_1‘..‘x_58‘
(covariates, mix of continuous and categorical). Ground-truth columns removed.

Task: Implement a deterministic estimator returning (ite_hat, ate_hat,
method_name).

Evaluation: source pool of 200 replicates (5 DGPs x 40 simulations);
main study uses a 100-replicate working subset per run.
Score = log(1+mean_sqrt_PEHE) + lambda*log(1+mean_ATE_err).
The combined score reported to OpenEvolve is 1/(1+score).

Objective: Minimise score. Explore diverse estimator families. Handle non-numeric
covariates robustly.

Requirements: Pure function. pandas, numpy, scikit-learn, statsmodels only.
Modify only the EVOLVE-BLOCK.

[Data preview appended at runtime]
```

LaLonde Evolution Prompt

```
You are an expert causal inference researcher and Python programmer working on
the LaLonde observational dataset (Dehejia-Wahba / PSID controls).

Dataset schema: ‘treatment‘ (binary), ‘y_factual‘ (re78 earnings, $0-$107K),
covariates: age, education, black, hispanic, married, nodegree, re74, re75.
Ground-truth columns y0, y1, and ite exist in the benchmark files but are
removed before candidate execution.

Domain guidance:
- Observational study with strong selection bias between groups.
- Prior earnings (re74, re75) are strongest predictors.
- Consider propensity-score trimming and overlap enforcement.
- Do not assume ATE is positive.

Task: Implement a deterministic estimator returning (ite_hat, ate_hat,
method_name).

Evaluation: 100 semi-synthetic replicates.
Score = log(1+normalized_sqrt_PEHE) + lambda*log(1+normalized_ATE_err).
The combined score reported to OpenEvolve is 1/(1+score).

Requirements: Pure function. pandas, numpy, scikit-learn, statsmodels only.
Modify only the EVOLVE-BLOCK.

[Data preview appended at runtime]
```

A.1.2 S1.2. Algorithmic workflow

Algorithm 1 in the main text summarizes the common search loop used across all 96 runs in the factorial experiment; the fixed hyperparameters, dataset splits, and scoring rules are detailed below.

Iterative mutation prompts. After the seed program, OpenEvolve used its built-in full-rewrite mutation template. Each prompt included the current parent program, current metrics and feature coordinates, previous attempts, top archived programs, sampled inspiration programs, and the benchmark-specific task constraints. The model returned a complete rewritten program with the same `estimate(...)` interface; complete prompt logs are archived with the evolution traces.

A.1.3 S1.3. Full hyperparameter configuration

Supplementary Table 1 lists the fixed settings and factorial factors used throughout the full benchmark sweep.

Supplementary Table 1: **Complete hyperparameter configuration.** Parameters held constant across the 96-run factorial experiment unless noted. Factorial factors indicated with †.

Parameter	**Value**
Evolutionary search	
Maximum iterations	100
Population size (MAP-Elites)	60
Number of islands	4
Archive size per island	25
Elite selection ratio	0.3
Exploitation ratio	0.7
Allow full rewrites	True
Diff-based evolution	False
Number of top programs in prompt	3
Checkpoint interval	5 iterations
LLM configuration	
API provider	OpenRouter
Primary model†	GPT-5 or Gemini 3 Flash Preview
Primary model weight	0.80
Secondary model	Claude Sonnet 4
Secondary model weight	0.20
Temperature	0.7
Top-p	0.95
Max output tokens	8,192
API timeout	600 s
Evaluation pipeline	
Cascade evaluation	True
Stage 1 threshold (θ_1)	0.001
Parallel workers (stage 2 search)	1
Parallel workers (final held-out evaluation)	4
Evaluation timeout per program	1,800 s
Dataset-specific	
ACIC 2022: training / held-out replicates	75 / 75 (50:50) or 30 / 120 (20:80), from a 150-replicate working subset sampled from 300 total
ACIC 2022: target coverage	0.90
IHDP: training / held-out replicates	12 / 13 (50:50) or 5 / 20 (20:80), from a 25-replicate working subset sampled from 50 total
ACIC 2016: training / held-out replicates	50 / 50 (50:50) or 20 / 80 (20:80), from a 100-replicate working subset sampled from 200 total
LaLonde: training / held-out replicates	25 / 25 (50:50) or 10 / 40 (20:80), from a 50-replicate working subset sampled from 100 total
Factorial factors†	
Datasets	ACIC 2022, ACIC 2016, IHDP, LaLonde
LLM (primary)	GPT-5, Gemini 3 Flash Preview
Regularization λ	0.2 (small), 1.0 (moderate), 5.0 (large)
Train–test split	50:50, 20:80
Independent replicates per cell	2
Reproducibility	
Deterministic replicate seeds	20250215, 20250216
Allowed Python packages	pandas, numpy, scikit-learn, statsmodels

A.1.4 S1.4. Composite scoring functions

Each benchmark uses a scalarized score so that MAP-Elites selection can trade off primary accuracy against secondary objectives such as ATE calibration or interval coverage.

ACIC 2022 (panel data): $S_{\text{ACIC}} = 1/(1 + \log(1 + \text{RMSE}) + \lambda \cdot \log(1 + |\text{Coverage} - 0.9|/0.9))$.

IHDP, ACIC 2016, LaLonde (cross-sectional): $S_{\text{ITE}} = 1/(1 + \log(1 + \overline{\sqrt{\text{PEHE}}}) + \lambda \cdot \log(1 + \overline{|\text{ATE error}|}))$, where the overlines denote averages across the training replicates evaluated in each iteration and $\sqrt{\text{PEHE}}$ denotes the unit-level ITE RMSE defined in Methods. The three levels of λ—0.2 (small), 1.0 (moderate), and 5.0 (large)—control the relative weight of secondary metrics.

A.1.5 S1.5. Implementation and reproducibility details

The notes below summarize the runtime environment and randomization controls used for the full experiment.

Software. The framework is implemented in Python 3.11 using OpenEvolve (Sharma, 2025) v0.2. LLM API calls are routed through OpenRouter. Candidate programs are executed in isolated subprocesses with enforced timeouts.

MAP-Elites archive. Each island used archive size 25 and population size 60. Bins were defined by OpenEvolve's built-in `complexity` and `diversity` feature dimensions, using 10 bins per dimension and a diversity reference set of 20 programs. Complexity is code length, and diversity is OpenEvolve's fast code-diversity score against the reference set.

Reproducibility. Two deterministic replicate seeds (20250215 and 20250216) control benchmark replicate sampling across the factorial grid. Per-run configuration files and evolved programs at every checkpoint are archived in the code repository.

LLM call counts and stylized API cost. Each evolution iteration issues one candidate-generation LLM call, so a 100-iteration run corresponds to 100 generation calls. Under the 0.80/0.20 primary/secondary ensemble weights, the expected composition is 80 primary-model calls and 20 Claude Sonnet 4 calls per run. The 96-run true-score factorial therefore corresponds to about 9,600 generation calls in total (expected 7,680 primary and 1,920 Claude calls); including the 48 matched proxy runs raises the total to about 14,400 calls. If a run uses on average b_{in} prompt tokens and b_{out} completion tokens per generation call, then its expected token budget is $100\, b_{\text{in}}$ input tokens and $100\, b_{\text{out}}$ output tokens, and the corresponding API spend is obtained by multiplying those totals by the provider's then-current model-specific per-token prices. For a representative bookkeeping example of 12k input and 2k output tokens per call, this corresponds to about 1.2M input tokens and 0.2M output tokens per run, about 115M/19M tokens over the 96-run main factorial, and about 173M/29M tokens when the 48 proxy runs are included.

A.2 S2. Additional benchmark summaries and complexity analyses

This section collects the supporting tables and figures referenced from the Results section. We first report the full zero-shot benchmark comparisons, then document the exact program-analysis inputs and prompts, summarize code similarity across final evolved programs, and finally show the expanded code-length and methods-length figures that sit behind the main-text complexity discussion.

A.2.1 S2.1. Full zero-shot benchmark tables

Supplementary Tables 2 and 3 provide the complete zero-shot comparisons referenced in the ACIC 2022 and IHDP results discussion. They contextualize the seed programs from which the evolutionary search began.

Supplementary Table 2: **Full zero-shot results on ACIC 2022 (50 replicates).** Seven LLMs evaluated under identical conditions. Each model was given the same structured prompt containing the data schema, causal assumptions, evaluation criteria, and a six-row data preview. Models were allowed up to five attempts to produce a valid program passing a 180 s warm-up test on a single replicate.

Model	Attempts	RMSE	Coverage	Method
Claude Sonnet 4	2	89.24	0.12	Weighted AIPW w/ RF
GPT-5	1	33.05	0.40	Patient-weighted TWFE DiD
GPT-4o	1	39.21	0.36	DiD + regression adjustment
Grok 4 Fast	3	89.24	0.22	Diff-in-means w/ bootstrap
Qwen3-Coder	1	68.93	0.30	Weighted TWFE (clustered)
DeepSeek v3.1	2	19.44	0.02	Weighted FE w/ clustered SEs
Gemini 2.5 Pro	5	—	—	*No valid program*

Supplementary Table 3: **Full zero-shot results on IHDP (50 replicates).** Seven LLMs evaluated under identical conditions. Means ± SE are computed across 50 replicates with ground-truth potential outcomes.

Model	Attempts	$\sqrt{\text{PEHE}}$	|ATE error|
Claude Sonnet 4	1	2.40 ± 0.55	0.103 ± 0.013
GPT-5	1	5.68 ± 0.73	0.292 ± 0.030
GPT-4o	1	1.86 ± 0.38	0.104 ± 0.014
Grok 4 Fast	1	1.86 ± 0.38	0.106 ± 0.014
Qwen3-Coder	2	3.11 ± 0.73	0.489 ± 0.141
DeepSeek v3.1	2	2.02 ± 0.34	0.103 ± 0.018
Gemini 2.5 Pro	5	—	—

A.2.2 S2.2. Program-analysis inputs and prompt summaries

This subsection documents the code pool and prompt templates behind the program-analysis results in Fig. 3, Table 3, and the complexity figures below. We analyzed one final code snippet per run: 28 zero-shot baselines, 96 true-evolved finals across both split regimes, and 48 matched proxy-evolved finals from the 20:80 proxy analysis. All embedding-based figures and tables in the main text use OpenAI `text-embedding-3-large`. We reran the same within- versus between-dataset and nearest-reference summaries with `Qwen3-Embedding-0.6B` as a sensitivity analysis; the alternative embedding space was more compressed, but it preserved the same qualitative ordering and dataset-level conclusions.

Supplementary Table 4: **Program-analysis inputs and prompt summaries.** Exact code sources and shared prompt templates for the program-analysis pipeline.

Component	Models / inputs	Retrieval rule or shared prompt
Program pool	28 baselines, 96 true-evolved finals, 48 proxy-evolved finals	One final snippet per run from the zero-shot baseline pool, the true-score final-program pool, and the proxy-score final-program pool. The exact 172-program manifest is archived alongside the analysis outputs.
Translation	GPT-5, Gemini 2.5 Pro, Claude Sonnet 4.5	Shared system and user template shown below; each prompt supplied the dataset name, source label, and full program text, and asked for a single manuscript-style methods paragraph.
Judge panel	GPT-5, Gemini 2.5 Pro, Claude Sonnet 4.5	Shared JSON-returning review template shown below; each prompt asked for family assignment, nearest known method, complexity, pipeline steps, and novel components before cross-model aggregation.
Embedding sensitivity	OpenAI main; Qwen sensitivity	All manuscript figures and tables use `text-embedding-3-large`. The same nearest-reference and within/between-program summaries were rerun with `Qwen3-Embedding-0.6B` only as a sensitivity analysis.

For the official reference library, we also archived the exact commit-pinned upstream example links used to build the seven normalized wrappers (see Supplementary Table 5).

Supplementary Table 5: **Commit-pinned upstream sources for the official reference library.** Each normalized `estimate(df)` wrapper used in the reference-matching analysis was adapted from the corresponding maintainer example at the listed commit-pinned URL.

Family	Wrapper	Package	Commit-pinned upstream example	Commit
T-learner	T-Learner (EconML TLearner)	EconML	https://github.com/py-why/EconML/blob/b0404c02be5af1f9a2cbc2bd66d9b47e38c6ebb4/econml/metalearners/_metalearners.py#L54-L62	b0404c0
X-learner	X-Learner (EconML XLearner)	EconML	https://github.com/py-why/EconML/blob/b0404c02be5af1f9a2cbc2bd66d9b47e38c econml/metalearners/_metalearners.py#L328-L336	b0404c0
DR-learner	DR-Learner (EconML DRLearner)	EconML	https://github.com/py-why/EconML/blob/b0404c02be5af1f9a2cbc2bd66d9b47e38c6ebb4/econml/dr/_drlearner.py#L542-L555	b0404c0
IPW	IPW (zEpid IPTW)	zEpid	https://github.com/pzivich/zEpid/blob/16a0f96f8b2c65df8715085801f21757d14 zepid/causal/ipw/IPTW.py#L85-L88	16a0f96
OLS	OLS (statsmodels OLS)	statsmodels	https://github.com/statsmodels/statsmodels/blob/184e7f8e5725613ba64cbabc9bf0a33354e1d95e/examples/python/ols.py#L31-L37	184e7f8
AIPW	AIPW (zEpid AIPTW)	zEpid	https://github.com/pzivich/zEpid/blob/16a0f96f8b2c65df8715085801f21757d14 zepid/causal/doublyrobust/AIPW.py#L74-L77	16a0f96
Causal forest	CausalForest (EconML CausalForestDML)	EconML	https://github.com/py-why/EconML/blob/b0404c02be5af1f9a2cbc2bd66d9b47e38c6ebb4/econml/dml/causal_forest.py#L551-L558	b0404c0

Code-to-Methods Translation Prompt

````
System:
You convert Python estimation code into a methods paragraph for a scientific
manuscript. Describe the statistical procedure, preprocessing, weighting,
model structure, effect-estimation target, and uncertainty quantification when
present. Describe only what the code actually implements. Do not speculate, do
not mention absent steps, and do not add commentary such as "the code does not
explicitly..." or "there is no..." unless the omission is itself an explicit
design choice encoded in the program. Avoid line-by-line narration, code
syntax, bullet points, or implementation trivia that would not belong in a
methods section. Return plain text only.

User:
Dataset: <dataset>
Program source: <source label>

Translate the following Python code into a single publication-style methods
paragraph.

```python
<full program inserted here>
```
````

LLM-as-a-Judge Aggregation Prompt

````
You are a causal inference expert reviewing a Python program that estimates
individual treatment effects (ITE) and average treatment effects (ATE).

Analyze this program and answer the following questions. Be precise and concise.

```python
<full program inserted here>
```

Respond in EXACTLY this JSON format (no markdown, no extra text):
{
  "algorithm_family": "<primary method family: one of T-Learner, S-Learner,
    X-Learner, DR-Learner, AIPW, IPW, OLS, CausalForest, Matching, Ensemble,
    Other>",
  "sub_methods": ["<specific sub-methods used>"],
  "most_similar_known_method": "<single most similar published method>",
  "similarity_to_known": <integer 1-10>,
  "novel_components": ["<algorithmic innovations not in the standard method>"],
  "novelty_explanation": "<1-2 sentence explanation>",
  "base_learners": ["<ML models used>"],
  "pipeline_steps": ["<ordered algorithm steps>"],
  "complexity_rating": <integer 1-10>
}
````

A.2.3 S2.3. Code similarity among final programs

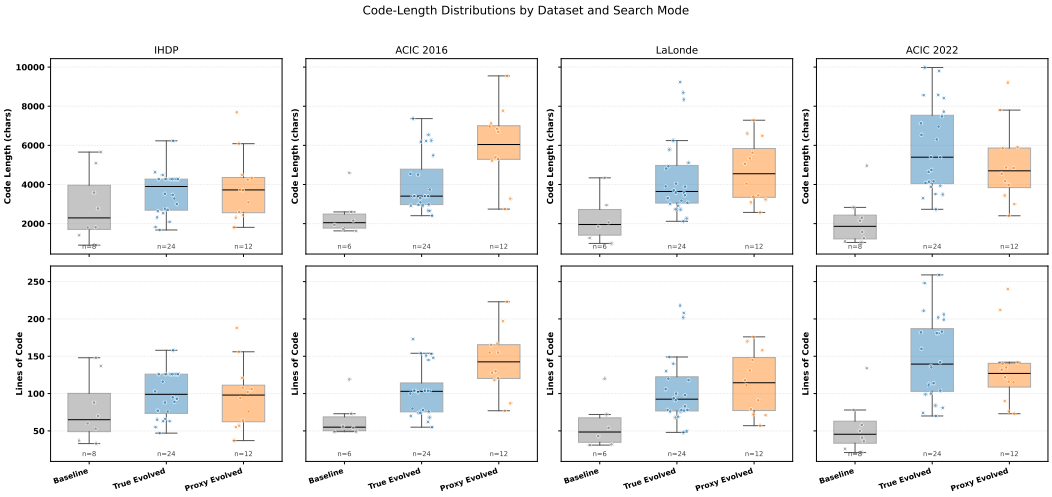
Supplementary Table 6 complements the main-text method-family summary by quantifying within- and between-dataset lexical similarity across the final evolved programs.

Supplementary Table 6: **Code similarity among final programs.** TF-IDF cosine similarity between pairs of final evolved programs. Within-dataset similarity is consistently higher than between-dataset similarity, confirming genuine dataset-specific adaptation.

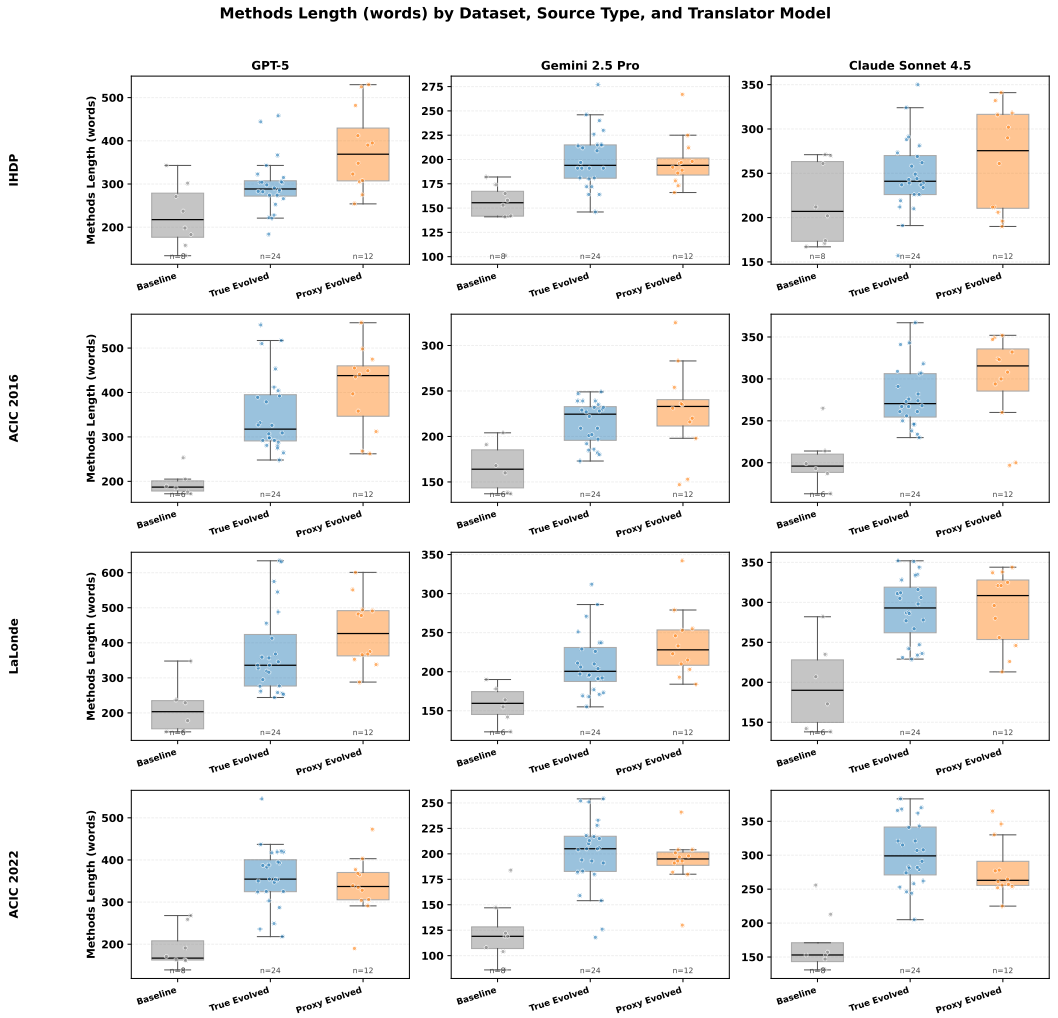
Comparison	Mean $\pm$ SD	n pairs
Within ACIC 2022	0.336 ± 0.110	276
Within ACIC 2016	0.353 ± 0.173	276
Within IHDP	0.415 ± 0.244	276
Within LaLonde	0.331 ± 0.116	276
ACIC 2022 vs. IHDP	0.140 ± 0.053	576
ACIC 2016 vs. IHDP	0.352 ± 0.186	576
ACIC 2022 vs. LaLonde	0.148 ± 0.063	576

A.2.4 S2.4. Program and methods-length summaries

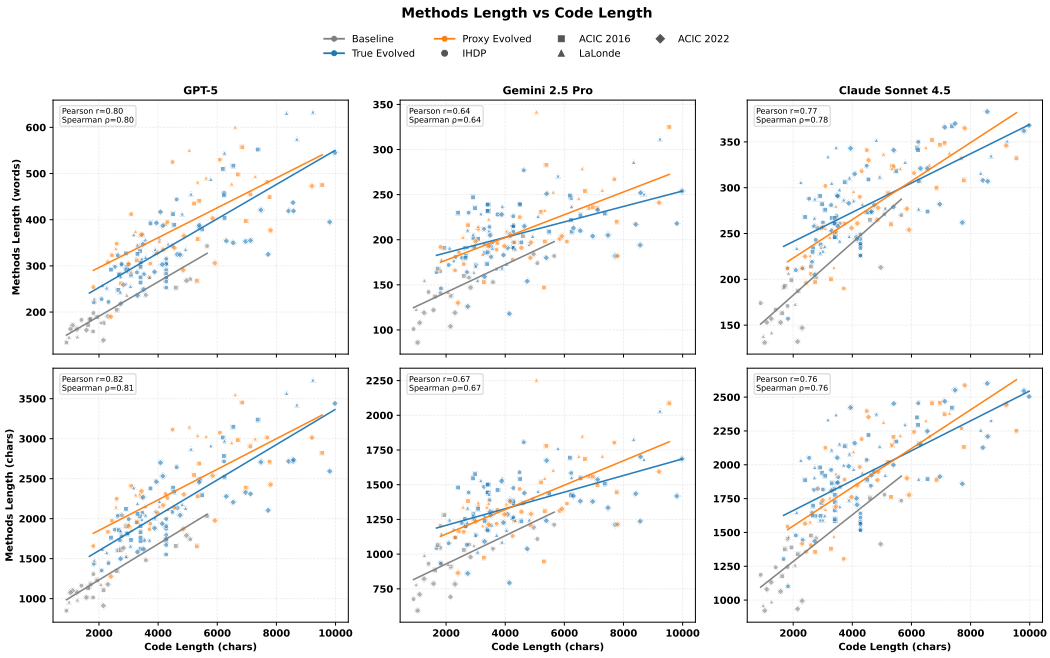
Supplementary Figs. 1, 2, 3, and 4 expand the compact complexity panels in Fig. 2. They show the full distributions of source-code length and manuscript-style methods length across datasets, translator models, and evolutionary settings.



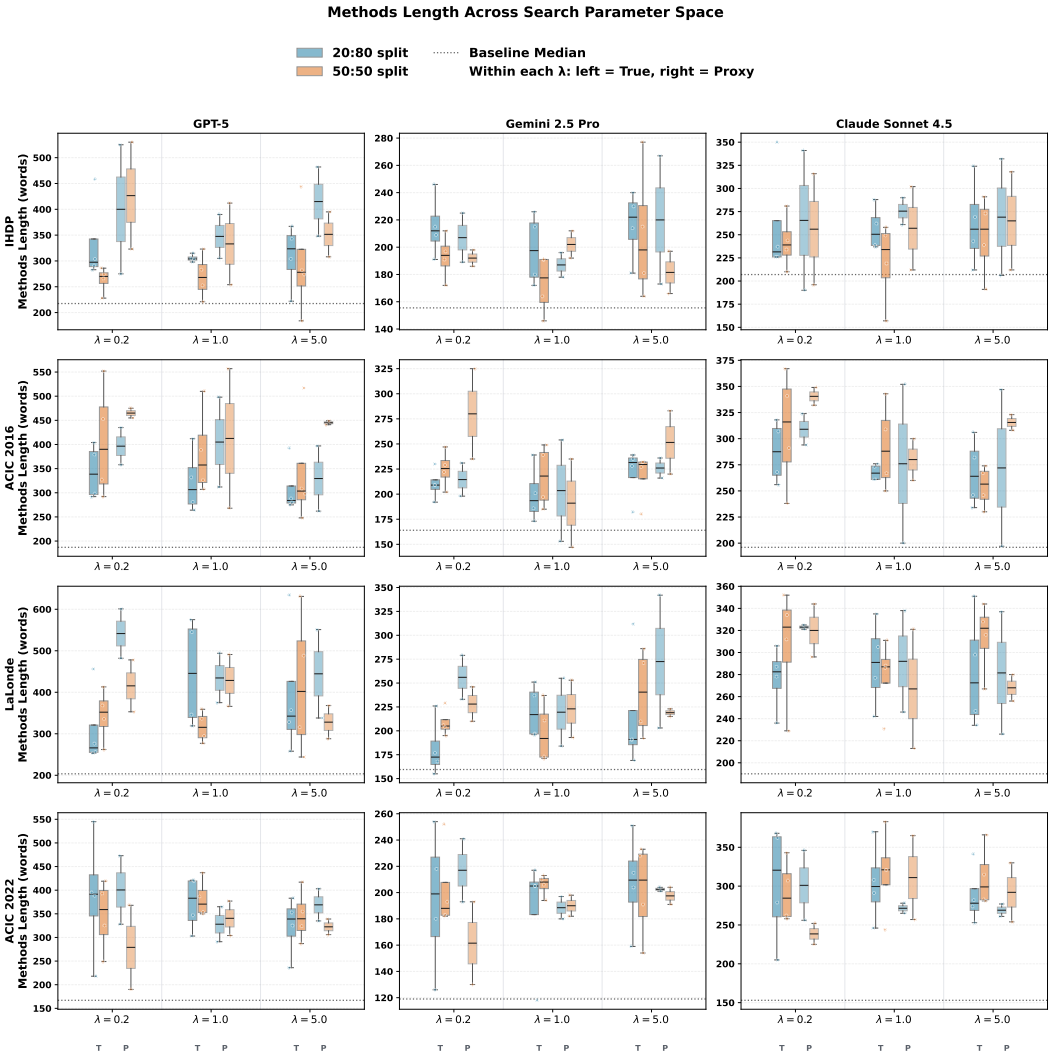
Supplementary Fig. 1: **Code length distributions across datasets and program sources.** Top panels show character count and bottom panels show non-empty lines of code for baseline, true-evolved, and proxy-evolved programs. Across all four benchmarks, evolved programs are consistently longer than the compact baseline programs, with the largest expansions occurring for ACIC 2022 true-evolved code and ACIC 2016 proxy-evolved code.



Supplementary Fig. 2: **Lengths of manuscript-style methods paragraphs derived from code.** Each program was translated into a single scientific-manuscript methods paragraph by GPT-5, Gemini 2.5 Pro, and Claude Sonnet 4.5 using the same unconstrained instruction template. Across datasets and translator models, baseline programs yield the shortest descriptions, while true-evolved and proxy-evolved programs produce substantially longer methods paragraphs.



Supplementary Fig. 3: **Translated methods length scales with source-code length.** Each point is one code-to-method translation. Longer source programs tend to yield longer manuscript descriptions across all three translator models, indicating that structural complexity discovered during evolution is reflected in both code size and textual methodological detail.



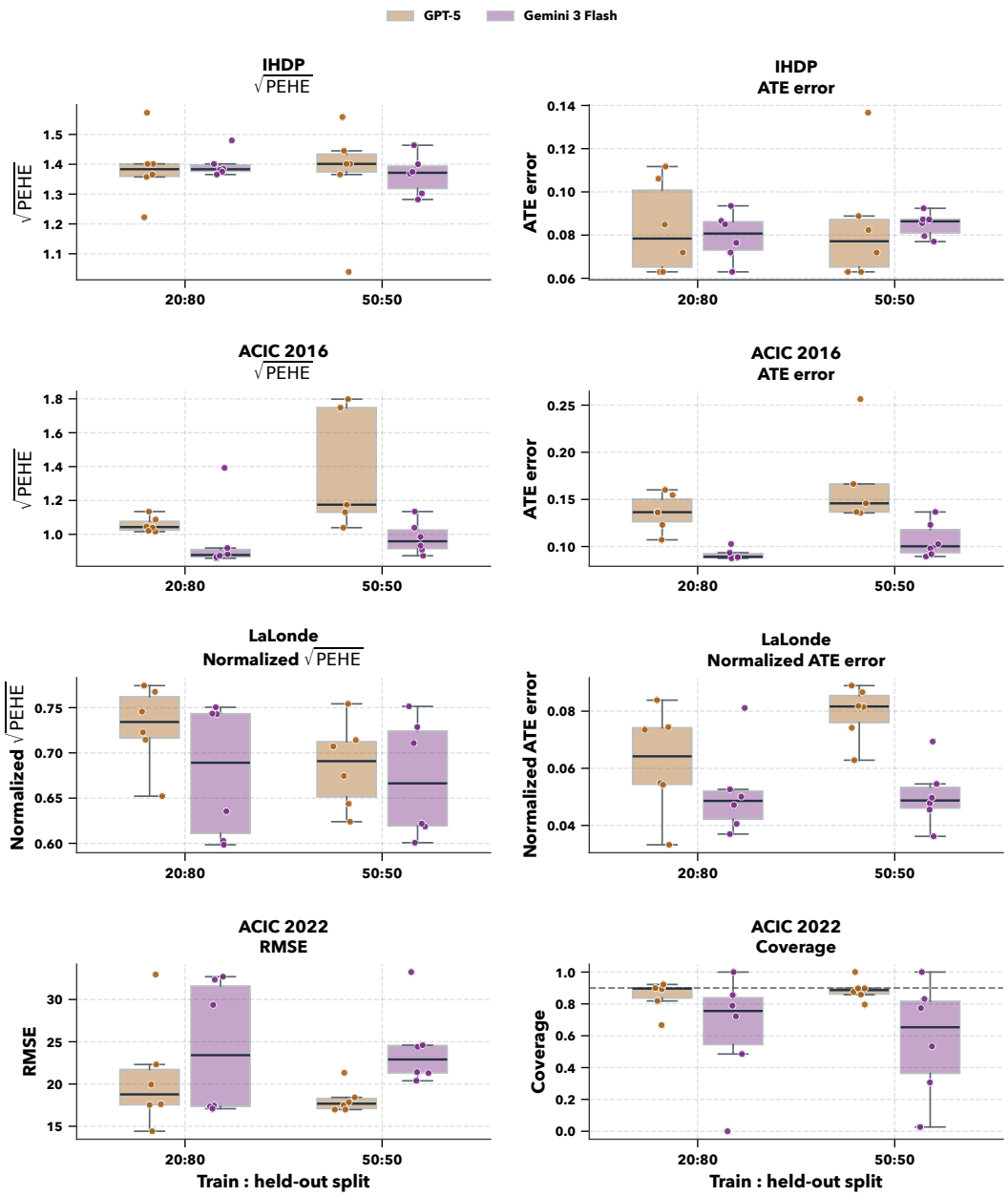
Supplementary Fig. 4: **Methods-paragraph length across the evolutionary parameter space.** Length distributions are stratified by dataset, translator model, source class, train–test split, and regularization level. Within each λ group, the left pair shows true-evolved programs and the right pair shows proxy-evolved programs; box color denotes the 20:80 versus 50:50 split. Source effects dominate split and regularization effects, although GPT-5 proxy-derived descriptions are longest under small λ.

A.3 S3. Sensitivity and proxy evaluation details

This section collects the supporting analyses behind the design-sensitivity and proxy-evaluation discussion in the Results section. We first document the held-out split sensitivity that is intentionally kept out of the main text, then decompose the regularization-weight comparison into its component metrics, and finally give the technical details and validation summaries for the proxy objectives.

A.3.1 S3.1. Held-out split sensitivity

All main-text figures and tables use the prespecified 20:80 split. Supplementary Fig. 5 shows the matched 50:50 reruns for true-score evolution, pooling over regularization settings and replicate seeds within each split–model cell. Across datasets, the held-out metric distributions are similar under 20:80 and 50:50, with split-to-split shifts smaller than the benchmark-level differences emphasized in the main text. We therefore use 20:80 as the single anchor split in the Results section and treat 50:50 as a robustness check.

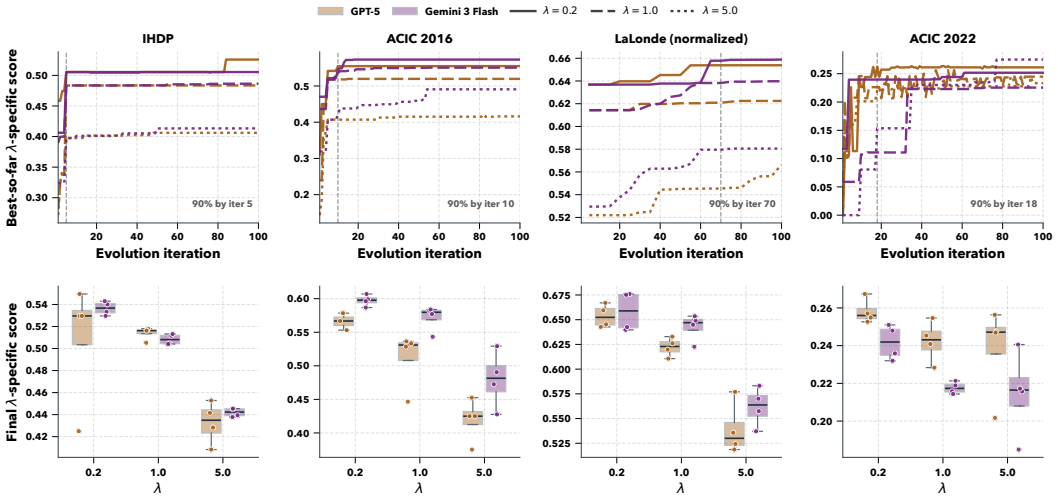


Supplementary Fig. 5: **Held-out sensitivity to the training split.** Each panel compares the pre-specified 20:80 main-analysis runs with matched 50:50 reruns, pooling over regularization settings and replicate seeds within each split–model cell; box color denotes the primary model. Columns show the held-out metrics reported in the main text and tables: $\sqrt{\text{PEHE}}$ and ATE error for IHDP and ACIC 2016, normalized $\sqrt{\text{PEHE}}$ and normalized ATE error for LaLonde, and RMSE plus empirical coverage for ACIC 2022. The dashed line in the ACIC 2022 coverage panel marks nominal 90% coverage. The similar distributions across split settings motivate anchoring the main text on 20:80 alone.

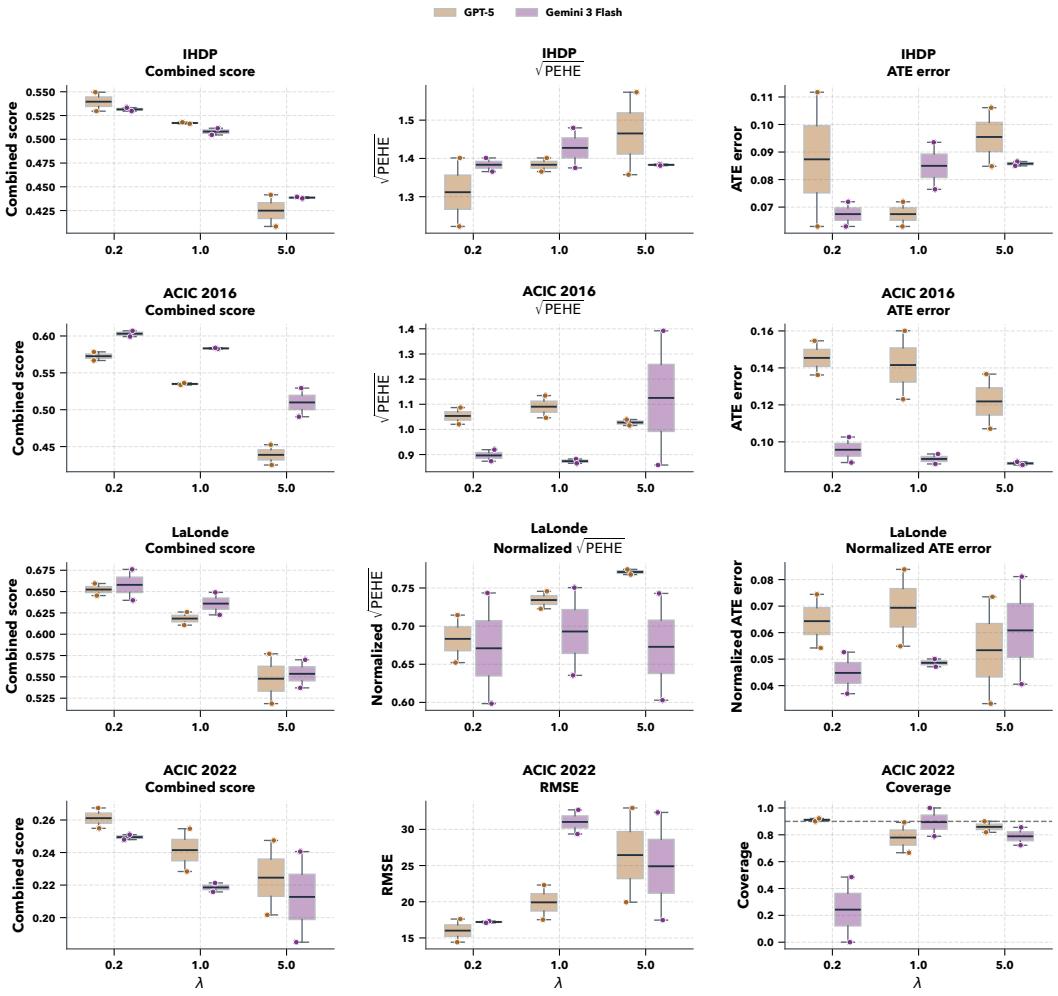
A.3.2 S3.2. Search dynamics and sensitivity to the regularization weight λ

Supplementary Fig. 6 separates the two primary models, GPT-5 and Gemini 3 Flash Preview, and shows that the effect of λ is broadly similar across them. Smaller λ generally favored lower held-out $\sqrt{\text{PEHE}}$ and/or ATE error on IHDP, ACIC 2016, and normalized LaLonde, whereas ACIC 2022 was comparatively flat over the tested range.

Supplementary Fig. 7 expands the final-score panel of Supplementary Fig. 6 into the combined score and the two component metrics used in each benchmark-specific objective, again under the main 20:80 split. Because λ enters the scalarized score definition, only the component-metric columns support direct cross-λ comparison of estimator quality.



Supplementary Fig. 6: **Search dynamics across models and λ.** Top: median best-so-far scores over 100 iterations for 20:80 runs, stratified by model and λ; dashed lines mark the median iteration reaching 90% of the final best-so-far score. Bottom: final search-score distributions, comparable within but not across λ settings.



Supplementary Fig. 7: **Sensitivity to λ across the combined score and its component metrics.** Each panel pools over replicate seeds within a dataset, λ, and primary-model cell under the main 20:80 split, with box colors denoting the primary model. Columns show the combined score, the primary accuracy metric, and the secondary metric used in the scalarized objective. Because λ appears in the combined-score definition, the first column should be interpreted as a λ-specific search objective rather than a directly comparable cross-λ performance scale; the second and third columns provide the valid held-out metric comparison. For IHDP, ACIC 2016, and LaLonde these component metrics are $\sqrt{\text{PEHE}}$ and ATE error (normalized for LaLonde); for ACIC 2022 they are RMSE and empirical coverage. The dashed line in the ACIC 2022 coverage panel marks nominal 90% coverage.

A.3.3 S3.3. DR pseudo-outcome PEHE proxy

For heterogeneous treatment effect estimation (IHDP, ACIC 2016, LaLonde-Norm), we construct cross-fitted doubly robust (DR) pseudo-outcomes following Kennedy (2024):

1. Split the data into $K = 2$ folds.
2. For each fold k, fit nuisance models on the complement $\{1, \ldots, K\} \setminus k$: outcome models $\hat{\mu}_t(x) = \mathbb{E}[Y \mid X = x, T = t]$ using both Ridge and histogram gradient-boosted regressors; propensity model $\hat{e}(x) = P(T = 1 \mid X = x)$ using logistic regression with clipping at $[0.05, 0.95]$; and a main-effect regression $\hat{m}(x) = \mathbb{E}[Y \mid X = x]$ using Ridge for the R-loss term.
3. Construct the DR pseudo-outcome for each unit i in fold k: $\tilde{\tau}_i = \hat{\mu}_1(x_i) - \hat{\mu}_0(x_i) + T_i(Y_i - \hat{\mu}_1(x_i))/\hat{e}(x_i) - (1 - T_i)(Y_i - \hat{\mu}_0(x_i))/(1 - \hat{e}(x_i))$.
4. Proxy ITE fit:

$$\widehat{\mathrm{RMSE}}_{\tau,\text{proxy}} = \left(N^{-1} \sum_i (\hat{\tau}_i - \tilde{\tau}_i)^2 \right)^{1/2}.$$

   Proxy ATE: $\hat{\tau}_{\text{AIPW}} = N^{-1} \sum_i \tilde{\tau}_i$.
5. We also compute a normalized R-loss term,

$$R_{\text{loss,norm}} = \frac{N^{-1} \sum_i \left(Y_i - \hat{m}(x_i) - (T_i - \hat{e}(x_i))\hat{\tau}_i\right)^2}{\widehat{\mathrm{Var}}(Y) + 10^{-8}}.$$

   The implemented ITE proxy objective is

$$\mathcal{L}_{\text{ITE-proxy}} = 0.5\, \phi(\overline{\widehat{\mathrm{RMSE}}_{\tau,\text{proxy}}}) + 0.3\, \lambda_{\text{ATE}}\, \phi(\overline{|\hat{\tau} - \hat{\tau}_{\text{AIPW}}|}) + 0.2\, \phi(\overline{R_{\text{loss,norm}}}),$$

   with $S_{\text{ITE-proxy}} = 1/(1 + \mathcal{L}_{\text{ITE-proxy}})$ and $\phi(x) = \log(1 + x)/(1 + \log(1 + x))$. As in Section S1.4, the overline denotes averaging across the training replicates used during search.

A.3.4 S3.4. DR-DID coverage proxy (ACIC 2022)

For ACIC 2022's panel data setting, we developed a DR difference-in-differences pseudo-target. We fit outcome models for control units in the pre and post periods, fit a propensity model for treatment assignment (clipped to $[0.05, 0.95]$), and construct the DR-DID estimate per replicate. The most reliable proxy signal was whether a candidate estimator's confidence interval covered this DR-DID target. To prevent trivial high-coverage solutions based on overly wide intervals, we also tracked raw interval width and, when enough replicates were available, a secondary width-calibration term

$$g_{\text{width}} = \frac{\left| \overline{W} - 2 z_{0.95}\, \widehat{\mathrm{sd}}(\hat{\tau}_{\text{DR-DID}}) \right|}{2 z_{0.95}\, \widehat{\mathrm{sd}}(\hat{\tau}_{\text{DR-DID}})},$$

which compares the mean reported interval width with the nominal 90% width implied by the cross-replicate spread of the DR-DID targets. The implemented objective is

$$\mathcal{L}_{\text{ACIC-proxy}} = 0.6\, \phi(\overline{|\hat{\tau} - \hat{\tau}_{\text{DR-DID}}|}) + 0.25\, \lambda_{\text{cov}} \widetilde{C} + 0.15\, \phi(\overline{W}),$$

where $\widetilde{C}$ is the hit-rate gap alone when too few replicates are available for calibration, and otherwise a 70:30 blend of hit-rate centering and cross-replicate width calibration. Specifically, if $\widehat{p}_{\text{hit}}$ is the fraction of evaluated training replicates whose candidate interval covers the DR-DID target, then

$$\widetilde{C} = \begin{cases} \phi(|\widehat{p}_{\text{hit}} - 0.9|/0.9), & R < 2, \\ 0.7\, \phi(|\widehat{p}_{\text{hit}} - 0.9|/0.9) + 0.3\, \phi(g_{\text{width}}), & R \geq 2, \end{cases}$$

where R is the number of evaluated training replicates. Across all 54 unique programs discovered across the 12 ACIC 2022 proxy runs, the DR-DID hit-rate component achieved $\rho = 0.955$ Spearman correlation with true coverage ($\rho = 0.99$ on an initial 76-checkpoint subset), and the DR-DID point-error component achieved $\rho = 0.859$ correlation with true SATT RMSE. Width calibration and raw interval width were retained in the scalarized objective as secondary regularizers against degenerate interval widths.

A.3.5 S3.5. Proxy validation and sensitivity analysis

Supplementary Table 7 reports the completed proxy-validation analysis across all 49 proxy-guided runs, evaluated at the unique-program level (every distinct best-so-far program discovered during search, $n = 243$ total across the four benchmarks) to avoid serial dependence across repeated checkpoints. For ACIC 2022, we report the coverage and point-estimation components separately because they represent qualitatively different statistical targets.

Supplementary Table 7: **Unique-program proxy metric validation across all 49 proxy-guided runs.** Each observation is one distinct best-so-far program discovered during proxy-guided evolution (243 unique programs in total). For IHDP, ACIC 2016, and normalized LaLonde, values report Pearson correlation r between proxy combined scores and ground-truth combined scores under matched λ. For ACIC 2022, values report Spearman rank correlation ρ split by coverage and point-estimation components.

Benchmark	Evaluation unit and component	n (unique programs)	Proxy vs. true correlation
IHDP	Unique program, combined score	55	Pearson $r = \mathbf{0.92}$
ACIC 2016	Unique program, combined score	69	Pearson $r = \mathbf{0.92}$
LaLonde	Unique program, combined score	65	Pearson $r = \mathbf{0.89}$
ACIC 2022 — coverage	Unique program, DR-DID hit-rate vs. true coverage	54	Spearman $\rho = \mathbf{0.96}$
ACIC 2022 — point est.	Unique program, DR-DID τ-error vs. true RMSE	54	Spearman $\rho = \mathbf{0.86}$

To assess sensitivity to the hyperparameters and functional form of the proxy objective itself, we re-executed and re-scored all 243 unique programs under perturbed proxy configurations varying one axis at a time from the prespecified defaults: cross-fitting fold count ($K \in \{2, 5, 10\}$), propensity-score clipping window ($[0.01, 0.99]$, $[0.02, 0.98]$, $[0.05, 0.95]$), outcome nuisance models (Ridge+GBM ensemble, Ridge-only, GBM-only), and pseudo-outcome construction (Supplementary Tables 8 and 9). Across all standard hyperparameter perturbations, proxy–truth correlations remained high and stable, whereas replacing the doubly robust DR-DID pseudo-outcome on ACIC 2022 with a naive DiD target sharply degraded both coverage ($\rho : 0.955 \rightarrow 0.67$) and point-estimation ($\rho : 0.859 \rightarrow 0.63$) alignment.

Supplementary Table 8: **Sensitivity of ITE proxy–truth correlations to proxy design choices.** Pearson correlation r between proxy and ground-truth combined scores across all unique discovered programs when perturbing cross-fitting folds K, propensity clipping bounds, or outcome nuisance learners one axis at a time.

Proxy variant (one axis changed from default)	IHDP ($n = 55$)	ACIC 2016 ($n = 69$)	LaLonde ($n = 65$)
Reported default ($K = 2$, clip $[0.05, 0.95]$, Ridge+GBM, AIPW)	**0.92**	**0.92**	**0.89**
$K = 5$ folds	0.89	0.91	0.82
$K = 10$ folds	0.87	0.91	0.82
Propensity clip $[0.01, 0.99]$	0.94	0.93	0.96
Propensity clip $[0.02, 0.98]$	0.94	0.92	0.94
Ridge-only outcome nuisances	0.95	0.94	0.95
GBM-only outcome nuisances	0.93	0.88	0.76
Range across all variants	**0.87–0.95**	**0.88–0.94**	**0.76–0.96**

Supplementary Table 9: **Sensitivity of ACIC 2022 DR-DID proxy correlations to proxy design choices and pseudo-outcome construction ($n = 54$ unique programs).** Spearman correlation ρ for the coverage and point-estimation components across proxy variants.

ACIC 2022 proxy variant	Hit-rate vs. true coverage (ρ)	τ-error vs. true RMSE (ρ)
Reported default (clip $[0.05, 0.95]$, in-sample outcome regression)	**0.955**	**0.859**
Propensity clip $[0.01, 0.99]$	0.957	0.864
Propensity clip $[0.02, 0.98]$	0.957	0.864
Cross-fitted outcome regression ($K = 2$ folds)	0.960	0.892
Gradient-boosted-tree propensity model	0.951	0.864
Pseudo-outcome set to naive DiD (non-doubly robust)	0.670	0.630

A.3.6 S3.6. Generalization levels and DGP split composition

Supplementary Table 10 summarizes the data-generating mechanisms of the four benchmarks and the generalization levels (Levels 1–2) established by our held-out and out-of-distribution evalua-

tions. For ACIC 2016 (200 total replicates across 5 DGP settings $\times$ 40 simulations), Supplementary Table 11 reports the exact realized DGP counts across the two deterministic replicate seeds used in the factorial sweep. Because splits were drawn by simple random sampling at the simulation level without explicit DGP stratification, all five settings appear on both sides of each split (with per-side goodness-of-fit χ^2, df $= 4, p \geq 0.15$ consistent with uniform random sampling), motivating the strictly DGP-stratified 50-dataset outer test reported in Table 1.

Supplementary Table 10: **Benchmark data-generating processes (DGPs) and generalization levels established by held-out evaluation.**

Benchmark	DGP specification and variation across replicates	Level established
IHDP	Nonlinear response surface, $\mu_0(x) = \exp((x+w)^\top \beta)$, $\mu_1(x) = x^\top \beta - \omega$, $Y(t) \sim \mathcal{N}(\mu_t(x), \sigma^2)$; covariates X and treatment T fixed while β, noise, and response surfaces vary per replicate	Levels 1 and 2
ACIC 2016	$T \sim \text{Bern}(\text{logit}^{-1}(f_T(X)))$, $Y(t) \sim \mathcal{N}(\mu_t(X), \sigma^2)$ with randomly generated additive f_T and μ_t; settings vary functional form, overlap, alignment, and treatment-effect heterogeneity	Levels 1 and 2 (plus 50-dataset outer test)
ACIC 2022	Clustered four-year practice/patient panel (DGP not public); 17 settings ($2 \times 2 \times 2 \times 2$ confounding and heterogeneity knob combinations plus one randomized setting), 200 datasets each	Levels 1 and 2
LaLonde–PSID	RealCause fits $p(W,T,Y)$ to LaLonde–PSID and samples $T, Y(0), Y(1)$; all replicates drawn from one fitted generative model with W fixed	Level 1

Supplementary Table 11: **Realized DGP composition of ACIC 2016 search and held-out splits.** Number of simulations from each of the five ACIC 2016 settings in the 100-replicate search set and its 100-replicate held-out complement for each deterministic seed.

Realized split	DGP 1	DGP 2	DGP 3	DGP 4	DGP 5
Search set, seed 20250215 ($n = 100$)	26	12	16	24	22
Held-out complement, seed 20250215 ($n = 100$)	14	28	24	16	18
Search set, seed 20250216 ($n = 100$)	14	25	18	24	19
Held-out complement, seed 20250216 ($n = 100$)	26	15	22	16	21

A.4 S4. Dataset-level evolution trajectories

The four subsections below summarize representative best-run trajectories for each benchmark. Each begins by restating the identification problem faced by the agent, then lists the key transitions along the winning run, and finally describes the resulting evolved estimator in prose.

A.4.1 S4.1. ACIC 2022 trajectory

Seed program and identification problem. ACIC 2022 presents a panel data problem: the agent must estimate a patient-weighted SATT from practice-year data with pre/post periods. The seed program is a TWFE-DiD estimator with post-year interactions and cluster-robust WLS, generated by GPT-5 in the zero-shot sweep (RMSE $= 33.1$, coverage $= 0.40$).

Key transitions. Supplementary Table 12 details the feature progression across all new-best checkpoints.

Final evolved program. The best ACIC 2022 program ("DR DiD-RA SATT, cluster-robust") computes per-practice pre-period weighted means and slopes, differences $Y_{\text{post}} - \bar{Y}_{\text{pre}}$, selects the top 8 covariates by correlation with control-group ΔY, runs per-year ridge-regularized WLS regression adjustment on controls, computes patient-weighted ATT on residuals, and estimates cluster-robust variance using influence-function contributions with conservative (max of residual vs. raw) variance selection and 5% SE inflation.

Supplementary Table 12: **ACIC 2022 feature progression.** Each row is a new-best checkpoint. The agent progressively discovers DiD logic, pre-mean adjustment, and trend correction.

Iter.	RMSE	Cov.	Ridge	Cov. enc.	Pre-mean	Trend	Post-only	CI adj.
0 (init)	33.1	0.40	—	—	—	—	—	—
2	31.7	0.55	✓	✓	—	—	—	—
14	28.7	0.55	—	✓	✓	—	—	—
16	27.1	0.75	—	✓	✓	—	—	✓
22	18.5	0.95	—	✓	✓	—	✓	✓
52	15.3	0.90	—	✓	✓	✓	✓	✓
88	15.2	0.90	—	✓	✓	✓	✓	✓

A.4.2 S4.2. ACIC 2016 trajectory

Seed program and identification problem. ACIC 2016 is a cross-sectional benchmark with 58 covariates and mixed continuous/categorical features. The seed program is an X-learner with cross-fitted gradient-boosted outcome models and constant propensity weighting.

Key transitions. Supplementary Table 13 follows a representative best-run trajectory, in which the agent refines the seed X-learner by replacing constant propensity weighting with a random-forest propensity model and local propensity-weighted blending.

Supplementary Table 13: **ACIC 2016 evolution checkpoints.** Best-run trajectory (Gemini, $\lambda = 0.2$, split 20:80). The agent refines a simple X-learner into a random-forest propensity-weighted X-learner with local blending.

Iter.	Method	$\sqrt{\text{PEHE}}$	|ATE error|	Combined
Init	X-Learner (GBM, cross-fitted)	1.225	—	baseline
Early	X-Learner + RF propensity (local weighting)	1.028	0.105	0.579
Final	X-Learner (GBM + RF PS)	1.028	0.105	0.579

Final evolved program. The best ACIC 2016 program ("X-Learner, GBM + RF PS") replaces the seed's constant propensity weighting with a random forest propensity model, enabling local propensity-weighted blending $\hat{\tau}(x) = \hat{e}(x)\hat{\tau}_0(x) + (1 - \hat{e}(x))\hat{\tau}_1(x)$. It uses `pd.get_dummies` to handle the 58 mixed-type covariates, 5-fold cross-fitting for outcome surfaces, and separate second-stage GBM models for treated and control pseudo-outcomes. The learning rate is tuned to 0.07 (vs. 0.05 in the seed). A separate high-performing GPT-5 variant, not the trajectory shown in Supplementary Table 13, evolved further into a "Meta-Ensemble (T+DR+DRF, adaptive)" that combines three base learners—T-learner, DR pseudo-outcome, and DR-smoothed random forest—using inverse-variance meta-weights from cross-validated performance.

A.4.3 S4.3. IHDP trajectory

Seed program and identification problem. IHDP is a semi-synthetic dataset with 25 covariates and known response surfaces. The seed program is identical to ACIC 2016: an X-learner with cross-fitted GBM outcome models and constant propensity weighting.

Key transitions. Supplementary Table 14 tracks a representative best-run trajectory, in which the agent augments the seed X-learner with an ensemble backbone, neural treatment-effect models, and adaptive propensity weighting.

Final evolved program. The best IHDP program ("TARNet-style X-learner + ensemble + adaptive-g") makes three key changes from the seed: (1) *Ensemble outcome models*: cross-fitted 70/30 blends of GBM and Ridge regression replace pure GBM, improving stability on the small IHDP samples. (2) *Neural network treatment effect models*: the second-stage $\hat{\tau}_1, \hat{\tau}_0$ models use MLPRegressor (hidden layers 50–25, L2 regularization $\alpha = 0.01$) with GBM fallback, capturing nonlinear heterogeneity. (3) *Feature-dependent propensity weighting*: rather than the constant $g = \bar{T}$ used in the seed, the program first computes a per-unit feature score $f_i = p^{-1} \sum_{j=1}^{p} |X_{ij}^{\text{scaled}}|$, the

Supplementary Table 14: **IHDP evolution checkpoints.** Best-run trajectory (GPT-5, $\lambda = 0.2$, split 20:80). The agent augments the X-learner with an ensemble backbone, neural network treatment effect models, and adaptive propensity weighting.

Iter.	Method	$\sqrt{\text{PEHE}}$	|ATE error|
Init	X-Learner (GBM, cross-fitted, constant g)	1.613	0.116
Early	GBM+Ridge ensemble $(0.7/0.3)$ + polynomial features	≈1.4	≈0.10
Mid	+ MLP treatment effect models (TARNet-style)	≈1.3	≈0.10
Final	TARNet-style X-learner + ensemble + adaptive-g	1.253	0.115

mean absolute value of the standardized covariates for unit i, then min-max normalizes f_i across units and sets $g_i = \text{clip}(g + 0.1(f_i - 0.5), 0.05, 0.95)$ for propensity-weighted blending of the X-learner arms. Additionally, the program adds interaction features between the top 5 scaled covariates and a propensity proxy, expanding the feature space from 25 to 30 dimensions.

A.4.4 S4.4. LaLonde trajectory

Seed program and identification problem. LaLonde is an observational job-training study with only 8 covariates but severe selection bias between experimental participants and survey-based controls. The seed program is the same X-learner used for IHDP and ACIC 2016.

Key transitions. Supplementary Table 15 follows a representative best-run trajectory in which the agent blends X-learner and R-learner components with AIPW calibration.

Supplementary Table 15: **LaLonde evolution checkpoints.** Best-run trajectory (GPT-5, $\lambda = 0.2$, split 20:80). The agent blends X-learner and R-learner components with AIPW calibration.

Iter.	Method	$\sqrt{\text{PEHE}}$	|ATE error|
Init	X-Learner (GBM, cross-fitted, constant g)	—	—
Early	+ Cross-fitted propensity model + DR pseudo-outcomes	≈1.5	≈0.20
Mid	X-learner + R-learner blend with AIPW	≈1.4	≈0.17
Final	CF X+R blend, AIPW-calib (GBR+Ridge)	1.359	0.161

Final evolved program. The best LaLonde program ("CF X+R blend, AIPW-calib, GBR+Ridge") is the most architecturally complex of the four datasets, reflecting the difficulty of observational bias correction. It combines:

- *Cross-fitted nuisance estimation*: 5-fold cross-fitting for outcome models (GBM, 160 trees, depth 3, subsample 0.8) and propensity scores (logistic regression with $C = 1.0$, clipped to $[0.02, 0.98]$).
- *X-learner component*: standard two-stage X-learner with GBM second-stage models and propensity-weighted blending $\hat{\tau}_X = \hat{e}\hat{\tau}_0 + (1 - \hat{e})\hat{\tau}_1$.
- *R-learner component*: residual-on-residual regression $Z_R = (Y - \hat{\mu}_{\text{mix}})/(T - \hat{e})$ with Ridge regression ($\alpha = 1.0$), weighted by $(T - \hat{e})^2$.
- *Ensemble blending*: 60% X-learner + 40% R-learner, calibrated so that $\bar{\hat{\tau}} = \hat{\tau}_{\text{AIPW}}$ via an additive shift.
- *Winsorization*: ITE predictions clipped at the 1st and 99th percentiles.

The evolution of an R-learner component is notable because the R-learner was not present in the seed program and represents a genuinely different identification strategy (residual-based) from the X-learner (imputation-based). The 60/40 blend weight and AIPW calibration were discovered through iterative mutation rather than being pre-specified.

A.5 S5. Representative code edits during evolution

This final section reproduces short diffs from major inflection points in the evolutionary runs. The goal is to make the trajectory descriptions above concrete by showing the kinds of code mutations associated with meaningful performance jumps.

Listings 1, 2, 3, 4, and 5 show representative mutations for ACIC 2022, IHDP, ACIC 2016, and LaLonde, respectively.

A.5.1 S5.1. ACIC 2022: Baseline → Iteration 2 (ridge regularization)

Listing 1: ACIC 2022: Ridge regularization and covariate encoding added.

```
@@
-import statsmodels.api as sm
+import statsmodels.api as sm
+from sklearn.preprocessing import StandardScaler
+from sklearn.linear_model import Ridge

@@
-    req = ['id.practice', 'year', 'Y', 'Z', 'post', 'n.patients']
-    for c in req:
-        if c not in df.columns:
-            raise ValueError(f"Missing required column: {c}")
+    d = df.copy()
+    covariate_cols = [col for col in d.columns if col.startswith(('X','V'))]
+    for col in covariate_cols:
+        if d[col].dtype == 'object' or d[col].dtype.name == 'category':
+            dummies = pd.get_dummies(d[col], prefix=col, drop_first=True,
+                                     dtype=float)
+            d = pd.concat([d, dummies], axis=1)
```

A.5.2 S5.2. ACIC 2022: Iteration 16 → 22 (post-period restriction)

Listing 2: ACIC 2022: Pivotal mutation retains pre-period anchoring and restricts estimation to post-period differences.

```
@@
-    post_years = sorted(d.loc[d['post']==1, 'year'].unique().tolist())
+    pre = d.loc[d['post'] == 0].copy()
+    pre_means = (pre.groupby('id.practice')
+                    .apply(lambda g: np.average(g['Y'],
+                                                weights=g['n.patients']))
+                    .rename('Y_pre_mean').reset_index())
+    d = d.merge(pre_means, on='id.practice', how='left')
+    post = d.loc[(d['post']==1) & np.isfinite(d['Y_pre_mean'])].copy()
+    post['D'] = post['Y'].astype(float) - post['Y_pre_mean'].astype(float)
```

A.5.3 S5.3. IHDP: Seed → best (ensemble outcome models + neural treatment effects)

Listing 3: IHDP: Seed X-learner evolves to TARNet-style ensemble.

```
@@
-from sklearn.ensemble import GradientBoostingRegressor
+from sklearn.ensemble import GradientBoostingRegressor
+from sklearn.linear_model import Ridge
+from sklearn.neural_network import MLPRegressor

@@  # Cross-fitted outcome models
-         m0 = GradientBoostingRegressor(**params)
-         m0.fit(X_train[control_idx], y_train[control_idx])
-         mu0_preds[test_idx] = m0.predict(X_test)
+         gbr0 = GradientBoostingRegressor(**gbr_params)
+         ridge0 = Ridge(alpha=1.0, random_state=42)
+         gbr0.fit(Xtr[c_mask], ytr[c_mask])
+         ridge0.fit(Xtr[c_mask], ytr[c_mask])
+         mu0_oof[te] = 0.7 * gbr0.predict(Xte) + 0.3 * ridge0.predict(Xte)

@@  # Second-stage treatment effect models
-     tau0 = GradientBoostingRegressor(n_estimators=100, ...)
-     tau0.fit(X_scaled[control_idx], d0[control_idx])
+     tau1_nn = MLPRegressor(hidden_layer_sizes=(50, 25), alpha=0.01, ...)
+     tau1_nn.fit(X_enhanced[t == 1], d1[t == 1])

@@  # Propensity weighting
-     g = np.mean(treatment)
-     ite_hat = g * tau1_pred + (1 - g) * tau0_pred
+     feature_weight = np.mean(np.abs(X_scaled), axis=1)
+     feature_weight = (feature_weight - feature_weight.min()) / (...)
+     adaptive_g = g + 0.1 * (feature_weight - 0.5)
+     adaptive_g = np.clip(adaptive_g, 0.05, 0.95)
+     ite_hat = adaptive_g * tau1_pred + (1 - adaptive_g) * tau0_pred
```

A.5.4 S5.4. ACIC 2016: Seed → best (local propensity weighting)

Listing 4: ACIC 2016: Constant propensity replaced by RF propensity model.

```
@@
-from sklearn.ensemble import GradientBoostingRegressor
+from sklearn.ensemble import GradientBoostingRegressor,
    RandomForestClassifier

@@  # Propensity score: global mean -> local RF estimate
-    g = np.mean(treatment)
-    g_pred = np.full(len(df), g)
-    ite_hat = g_pred * tau1_pred + (1 - g_pred) * tau0_pred
+    ps_model = RandomForestClassifier(n_estimators=100, max_depth=5,
+                                      min_samples_leaf=10, random_state=42)
+    ps_model.fit(X, t)
+    ps = np.clip(ps_model.predict_proba(X)[:, 1], 0.05, 0.95)
+    ite_hat = ps * tau0_x + (1 - ps) * tau1_x
```

A.5.5 S5.5. LaLonde: Seed → best (R-learner + AIPW calibration)

Listing 5: LaLonde: X-learner augmented with R-learner and AIPW calibration.

```
@@  # New: R-learner component (not in seed)
+    mu_mix = ehat * mu1 + (1 - ehat) * mu0
+    y_res = y - mu_mix
+    t_res = t - ehat
+    z_r = np.where(np.abs(t_res) > 1e-3, y_res / t_res, 0.0)
+    w_r = np.clip(t_res ** 2, 1e-4, None)
+    r_model = Ridge(alpha=1.0, random_state=42)
+    r_model.fit(Xs, z_r, sample_weight=w_r)
+    tau_r = r_model.predict(Xs)

@@  # New: Ensemble blend + AIPW calibration (not in seed)
+    ite_hat = 0.6 * tau_x + 0.4 * tau_r
+    shift = ate_dr - float(np.mean(ite_hat))
+    ite_hat = ite_hat + shift  # calibrate to AIPW ATE
```

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the `InferenceEvolve` framework, benchmark scope, proxy-objective contribution, and trajectory-analysis claims; the corresponding evidence appears in the Results and Methods sections.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The Discussion section describes the simulation-to-reality gap, proxy-objective limitations, computational cost and benchmark specialization, deployment boundaries, and future robustness directions.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper is empirical and methodological, with metric definitions and an algorithmic workflow but no theorem statements or formal proof claims.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The Methods and supplementary material describe datasets, splits, scoring functions, prompts, LLM settings, search hyperparameters, evaluation stages, and the repository location.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The Data availability and Code availability sections identify the public benchmark sources and the project repository with reproduction materials.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The Methods and supplementary Search setup sections specify benchmark splits, score definitions, model choices, search parameters, candidate-evaluation cascade, allowed libraries, and timeouts.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper reports distributions, mean $\pm$ SE summaries, coverage metrics, held-out evaluations, paired component metrics, and sensitivity analyses over splits, λ, and proxy objectives.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The current draft gives per-candidate timeouts, worker settings, and LLM-call settings.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The study uses public benchmark data and code-generation experiments and does not involve new human-subjects data collection, private data, or deployed decision systems.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: The Discussion summarizes positive accessibility and stress-testing benefits as well as risks from over-trusting or misusing generated estimators for unsupported causal claims.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper does not release high-risk pretrained models, image generators, scraped datasets, or other assets that require controlled-access safeguards.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The Data and Code availability sections credit the public benchmark sources, state that dataset and package licenses and terms will be documented with the release, and specify the intended CC BY 4.0 or compatible license for the de-anonymized code release.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: The paper releases code, prompts, evaluation scripts, and analysis workflows through the project repository, with supplementary descriptions of the benchmark setup and prompt templates.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A]

Justification: The reported experiments use public benchmark datasets and existing competition summaries; the paper does not conduct new crowdsourcing or new human-subjects data collection.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not conduct new human-subjects research or collect new participant data, so IRB approval is not applicable to the reported experiments.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLM use is central to `InferenceEvolve` and is described in the Methods section, including the model families, ensemble configuration, prompt templates, and evaluation loop.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.