# Off-policy Estimation in the Agentic Era

Max Pagels

July 21, 2026

*Disclaimer: This is not a research paper but rather the author's own thoughts. It does not claim originality, and is intended as a small study on apply know off-policy evaluation techniques to agents.*

## 1 Introduction

With the rise in utility of large language models (LLMs), a similarly accelerated rise in related tooling has taken place. Long-running agents have become commonplace, typically orchestrated by harnesses, which decompose large problems into a sequence of smaller, sequential decisions. Simple question-and-answer tasks have given way to complex chains of tool calls, retrieval steps, sub-agent delegations, and conditional branches, each conditioned on the history of what came before.

This shift has changed what it means to evaluate an LLM system. When a model's job was to answer a single question, evaluation was comparison against a reference: assemble a fixed set of prompts, score the outputs via an evaluator, and decide based on measured improvement. This can be thought of as a form of regression testing, and much of the tooling around agents continues to use it: LLMs and their harnesses, both for implementors and users, are regularly evaluated using golden datasets, LLMs-as-judges, gates in continuous integration pipelines, and so on. Implementors may perform these types of regression tests to ensure alignment, safety, operational cost effectiveness, and accuracy, whilst users of agentic products may optimise for different objectives.

Regression testing answers valid questions: do new versions of agents still pass the cases we have chosen to care about, under the constraints we have set? It does not answer the more fundamental, counterfactual question: if we were to deploy this change, what would happen?

Counterfactuals are expensive to answer, and one could argue more expensive than usual for demanding tasks. Running a live experiment via A/B testing might have been feasible when using an LLM meant posing simple questions, but now, the monetary cost of experimentation may be prohibitive and, as agents move into workflows where consequences are regulated or irreversible, other forms of cost may become too large to bear. In such settings, a trustworthy estimate obtained without significant live experimentation becomes a

precondition, not simply a convenience.

Off-policy evaluation—estimating the value of a policy you have not deployed, from data logged under a policy you did deploy—long predates the current interest in agents. Its derived estimators, and failure modes, have been extensively studied in contextual bandit and reinforcement learning literature over the past decades, yet we don't see widespread adoption of these techniques. This paper argues that the gap is not one solely of algorithms but also of their adaptation to the problem setting. Estimators exist and are well understood; what is missing is trust in those estimators to provide trustworthy and, crucially, *actionable* estimates specifically for agents. The paper makes two claims:

Claim 1: any application of off-policy evaluation of agents by importance-weighting entire trajectories of tokens is not merely difficult but intractable beyond a short horizon. A trajectory-level weight is a product of per-decision importance ratios, the variance of which grows exponentially using common estimators. It is unclear if amount of engineering can overcome this fundamental problem in a principled fashion.

Claim 2: the very nature of agentic workflows—a sequence of smaller decisions, conditioned on history—itself provides a solution to Claim 1, if we are willing to accept a relaxation of the stated aim of off-policy evaluation. If we evaluate one decision at a time, each choice is a single-factor off-policy problem with well-behaved variance which can be estimated using well-understood estimators. Such estimates are necessarily biased, as we shall see, but seem more usable in practice than any estimate under Claim 1.

## 2 Off-policy Evaluation for LLM Agents

Off-policy evaluation is tasked to determine, using data generated by a deployed logging policy $\pi_l$, how a candidate policy $\pi_c$ would have performed had it been used in place of $\pi_l$ during the time the logging policy generated its observations. This is the ideal target; as we will see in Section 4, what can actually be estimated in practice is a slightly weaker quantity, since the data only ever reflects the histories $\pi_l$ actually visited.

All policies are assumed to be stochastic: when they return an action $a$ to play in the environment, they do so by sampling from a probability distribution. We write $p_a^{\pi_l} = \pi_l(a \mid x)$ for the probability the logging policy assigned to action $a$ conditional on context $x$, and $p_a^{\pi_c} = \pi_c(a \mid x)$ for the probability the candidate policy would have assigned to the same action in the same context. The *context* $x$ is the information available at the moment of the decision: in an agentic setting, this is the history of everything observed so far—the prompt, all prior observations, actions, and tool results. The *reward* $r$ is a scalar measure of outcome quality attached to the logged observation, whether a per-step signal or a single terminal outcome; it is the quantity whose expectation under a policy defines that policy's value. Any action the candidate policy might take must have a nonzero logging probability $p_a^{\pi_l} > 0$—an overlap requirement that means non-greedy (temperature-nonzero) LLM generations must be used. These

probabilities are assumed to be recorded at decision time and made available to the offline evaluation algorithm; for the avoidance of any doubt, probability here means the value under the sampling distribution actually used, not the raw model distribution before temperature and truncation are applied.

What constitutes an action depends on the setting. For agents, $a$ might be the next token in sequence, with $p_a^{\pi_l}$ the probability by which it was chosen; an entire trajectory, with $p_a^{\pi_l}$ the probability of producing it; or a single decision within a turn, such as which tool to call. The three are not equivalent choices. A token probability is read off directly at decision time; a trajectory probability is not separately stored but derived, by multiplying the per-token probabilities along the trajectory; and a per-decision probability is either recorded directly, when the decision is made by an explicit stochastic selector, or derived from the tokens that constitute the action. The consequences of choosing trajectory-level actions are discussed in Section 3, where it is shown to be an impractical choice.

Once logged observations exist, offline evaluation iterates over them in the order by which they were recorded and applies an estimator that aims to correct for the non-uniformity by which actions were generated. The classic estimator is the inverse-propensity score (IPS) estimator, which weighs each logged observation by the ratio of the probability the candidate policy would have assigned to the action taken to the probability the logging policy actually assigned to it. For a log of $n$ observations, where observation $i$ records a context $x_i$, an action $a_i$ chosen by the logging policy, and a reward $r_i$, the estimator of the candidate policy's value is

$$\hat{V}_{\mathrm{IPS}}(\pi_c) = \frac{1}{n} \sum_{i=1}^{n} \frac{p_{a_i}^{\pi_c}}{p_{a_i}^{\pi_l}} \, r_i = \frac{1}{n} \sum_{i=1}^{n} \frac{\pi_c(a_i \mid x_i)}{\pi_l(a_i \mid x_i)} \, r_i, \tag{1}$$

where $p_{a_i}^{\pi_l} = \pi_l(a_i \mid x_i)$ is the logged propensity recorded at decision time and $p_{a_i}^{\pi_c} = \pi_c(a_i \mid x_i)$ is the probability the candidate policy assigns to the same action in the same context. The ratio $w_i = p_{a_i}^{\pi_c} / p_{a_i}^{\pi_l}$ is the importance weight: actions the candidate policy favours more than the logging policy did are up-weighted, and actions it favours less are down-weighted. IPS is famously unbiased: dividing by the logging propensity is what removes the bias introduced by logging under $\pi_l$ rather than $\pi_c$; under the overlap assumption above, the estimator is unbiased, so that $\mathbb{E}[\hat{V}_{\mathrm{IPS}}(\pi_c)] = V(\pi_c)$. Unbiasedness, however, comes at the cost of variance, and the variance can be severe: a single logged observation contributes a term $w_i r_i$ whose weight $w_i = p_{a_i}^{\pi_c} / p_{a_i}^{\pi_l}$ is inversely proportional to the logging propensity. When $p_{a_i}^{\pi_l}$ is small—a realistic concern especially in the LLM setting—the weight is correspondingly large, and a single such observation can dominate the entire estimate. Unbiasedness is a statement about the expectation over many hypothetical logs; a single log may still produce a poor estimate, and a single log is the only experience available in production settings.

The second estimator used in this paper is a modification of IPS that makes it self-normalising, at the expense of unbiasedness. Using self-normalising IPS

(SNIPS), the policy's value is estimated as

$$\hat{V}_{\mathrm{SNIPS}}(\pi_c) = \frac{\sum_{i=1}^{n} w_i\, r_i}{\sum_{i=1}^{n} w_i}, \qquad w_i = \frac{p_{a_i}^{\pi_c}}{p_{a_i}^{\pi_l}}, \tag{2}$$

where the sum of weighted rewards is divided by the sum of the weights rather than by the number of observations $n$. The change is small but its consequences are not. Under IPS, the divisor $n$ is a constant, and the weights are only required to average to one in expectation; in any particular finite log they do not, and it is precisely this discrepancy that allows an estimate to drift outside the range of observed rewards. SNIPS removes the discrepancy by construction: dividing by the realised sum of weights forces the estimate to be a weighted average of the observed rewards, and a weighted average of quantities lying in a given range must itself lie in that range. It is therefore impossible for SNIPS to produce an estimate that a single outcome could not theoretically have produced.

The price is bias. $\hat{V}_{\mathrm{SNIPS}}$ is a ratio of two quantities that are both random: the weighted-reward sum in the numerator and the sum in the denominator. The expectation of a ratio is not in general the ratio of expectations. SNIPS is therefore biased for any finite $n$. The bias is, however, of a benign kind: it vanishes as the log grows, so that $\hat{V}_{\mathrm{SNIPS}}(\pi_c) \to V(\pi_c)$ as $n \to \infty$, and it does so faster than the estimator's standard error shrinks, so that for logs of any practical size the bias is dominated by the variance it was introduced to control. In exchange for a bias that is small and disappearing, SNIPS delivers estimates that are bounded, far less sensitive to individual low-probability observations, and in practice far closer to the true value on the single log one actually has—which, as argued above, is the only log one ever gets.

# 3 The Horizon Wall (Claim 1)

Returning to Claim 1, this section shows that defining actions as entire agent trajectories in the LLM setting is intractable and, in practice, unusable for off-policy evaluation. Under this choice, the logged observations are whole trajectories: a trajectory $\tau$ of horizon $T$ is the sequence of actions $a_1, \ldots, a_T$ taken at histories $h_1, \ldots, h_T$, and the estimators of Section 2 are applied with each trajectory playing the role of a single logged action. Both IPS and SNIPS then require, per trajectory, the importance weight $w_\tau = p_\tau^{\pi_c}/p_\tau^{\pi_l}$; it is this weight whose behaviour decides whether either estimator is usable.

A trajectory's probability under a policy is not, however, due to the policy alone. Running an agent interleaves two stochastic actors: the policy, which samples each action, and the environment, which samples each response—a tool result, a retrieved document, a user reply. The probability of the trajectory is a product over both,

$$p_\tau^{\pi} \;=\; \prod_{t=1}^{T} \underbrace{\pi(a_t \mid h_t)}_{\text{policy}}\; \underbrace{P(o_t \mid h_t, a_t)}_{\text{environment}}, \tag{3}$$

where $P(o_t \mid h_t, a_t)$ is the environment's probability of producing response $o_t$. The environment responds to an action identically regardless of which policy produced it, so its factors are the same in $p_\tau^{\pi_c}$ and $p_\tau^{\pi_l}$ and cancel in their ratio. This is fortunate, since the environment's probabilities are typically unknown and could not be computed. What remains is a ratio of the policies' action probabilities alone, and for a trajectory this ratio is itself a product of the per-token ratios:

$$w_\tau \;=\; \frac{p_\tau^{\pi_c}}{p_\tau^{\pi_l}} \;=\; \prod_{t=1}^{T} \frac{\pi_c(a_t \mid h_t)}{\pi_l(a_t \mid h_t)} \;=\; \prod_{t=1}^{T} w_t, \qquad w_t = \frac{p_{a_t}^{\pi_c}}{p_{a_t}^{\pi_l}}. \tag{4}$$

This product is the single quantity that the trajectory-level IPS and SNIPS estimators of Section 2 depend on: $\hat{V}_{\mathrm{IPS}}$ averages $w_\tau r_\tau$ over logged trajectories, and $\hat{V}_{\mathrm{SNIPS}}$ forms the self-normalised ratio of the same terms. Everything that follows concerns the behaviour of $w_\tau$, since both estimators inherit it directly.

One might think it is the smallness of entire-trajectory probabilities that makes trajectory-level estimation unusable for off-policy evaluation in LLM settings: each is a product of per-token probabilities and so becomes infinitesimally small as the horizon grows. This may be true, but it is not the fundamental source of the problem. The estimators presented never use a trajectory probability on its own; they use the ratio $w_\tau$, and in a ratio the shrinkage of numerator and denominator partly cancels. A vanishingly small $p_\tau^{\pi_c}$ divided by a vanishingly small $p_\tau^{\pi_l}$ may be a perfectly ordinary number.

The problem is variance, and it is multiplicative. Each factor $w_t$ is a random quantity that scatters around one—by construction, $\mathbb{E}_{a_t \sim \pi_l}[w_t] = 1$, since summing $\pi_l(a_t \mid h_t) \cdot \pi_c(a_t \mid h_t)/\pi_l(a_t \mid h_t)$ over actions returns the total probability mass of $\pi_c$. The trajectory weight is the product of $T$ such factors, and the variance of a product grows with the number of factors multiplied. Treating the per-step ratios as independent for illustration, the variance of the product is

$$\mathrm{Var}(w_\tau) \;=\; \prod_{t=1}^{T} (1 + \mathrm{Var}(w_t)) \;-\; 1, \tag{5}$$

using $\mathbb{E}[w_t] = 1$. If each step contributes a comparable variance $\mathrm{Var}(w_t) = v > 0$, this is $(1+v)^T - 1$: any per-step variance, however small, is compounded $T$ times and the trajectory-weight variance grows exponentially in the horizon. Where the intuition of shrinking probabilities suggested a difficulty that a ratio might cancel, the variance of that ratio suggests a difficulty that nothing can: it is not that the weights are small, but that they are wildly unequal, a few enormous weights among a great many negligible ones. Fed into IPS, this scatters the estimate; fed into SNIPS, whose value is a weighted average dominated by its largest weights, it collapses the estimate onto the few trajectories that happen to carry almost all the weight, as explained later in Section 5.

# 4 Per-decision Decomposition (Claim 2)

Claim 2 states that the variance explosion of Claim 1 can be overcome by decomposing the trajectory into smaller constituent decisions, larger than a single token but smaller than a full trajectory across the entire task. As noted, such a solution is necessarily biased: each per-decision estimate is conditioned on the history that actually occurred under the logging policy, and so cannot capture how the candidate policy's own earlier choices would have steered it into different histories later on.

Consider the following decomposition: instead of treating a full trajectory as a sequence of token-level decisions, we decompose an entire task run into a sequence of $T$ decisions, where one decision—and thus one action—consists of a substring of the full trajectory's tokens.

A trajectory of horizon $T$ then contributes not one logged record but $T$ of them, each consisting of the history $h_t$ at which the decision was made, the action $a_t$ the logging policy took, its logged propensity $p^{\pi_l}_{a_t}$, and a reward attributable to that decision. The estimators of Section 2 are applied to this expanded log exactly as before, but now the importance weight of each record is a single per-step ratio $w_t = p^{\pi_c}_{a_t}/p^{\pi_l}_{a_t}$ rather than a product over the horizon, and the variance explosion of Claim 1 never takes place.

Because an agent trajectory typically yields a single outcome rather than a reward at every step, we attach that terminal outcome to each decision along the trajectory: a decision is judged by the reward $r$ of the trajectory it was part of. This is the precise point at which the bias of Claim 2 enters, since a decision is thereby credited with an outcome produced under the logging policy's continuation, not the candidate's.

**Composing a single estimate.** The decomposition raises an immediate question: having replaced one trajectory by $T$ per-decision records, and being able to weight each by its own ratio $w_t$, how are these combined into a single estimate of the candidate policy's value? Multiplication is out of the question, since that is what produced the variance explosion of Claim 1. Instead, the paper proposes a simple form of *pooling*. All per-decision records from all logged trajectories are placed into one collection, and a single estimator is computed over that collection. If the expanded log contains $M$ decision records in total, indexed by $j$, the per-decision IPS and SNIPS estimators are

$$\hat{V}^{\mathrm{dec}}_{\mathrm{IPS}}(\pi_c) = \frac{1}{M}\sum_{j=1}^{M} w_j\, r_j, \qquad \hat{V}^{\mathrm{dec}}_{\mathrm{SNIPS}}(\pi_c) = \frac{\sum_{j=1}^{M} w_j\, r_j}{\sum_{j=1}^{M} w_j}, \tag{6}$$

where each record $j$ carries a single-step weight $w_j = p^{\pi_c}_{a_j}/p^{\pi_l}_{a_j}$ and the reward $r_j$ attached to it. The composition is additive, not multiplicative: decisions contribute in parallel to one sum, never in series to one product. It is this that keeps the weights single-factor and the variance bounded, no matter how long the trajectories from which the decisions were drawn.

With the estimator fully assembled, it is worth stating plainly what quantity it targets, since it is not $V(\pi_c)$ as ideally off-policy evaluation would like to estimate. Every record corrects a single choice: the history leading up to it was produced by the logging policy, and the reward attached to it was produced by the logging policy's continuation. The pooled estimate therefore converges to what could be termed the *one-step deviation value*: the expected outcome of running the logging policy as usual, except that at one decision point along the way the candidate chooses instead, averaged over the decision points the logging policy visits. It measures, in other words, whether the candidate's judgement improves outcomes one decision at a time, within the situations the deployed system actually encounters—not what would happen if the candidate's choices were allowed to compound into situations of its own making. This relaxation may seem too severe, but the justification is empirical: if a new, proposed agentic system is not an improvement one decision at a time, it is unlikely to be an improvement when deployed in full. In other words, if you assume that on expectation, a new policy that is better than the old one in general if it is better than the old one in the situations the old one actually encounters, then one-step deviation value is a useful metric for pre-deployment evaluation.

# 5 Simulation: The Wall

Both claims in this paper are theoretical; the simulations in this section argue the claims empirically via simulation in the simplest possible environment, as failure modes in such an environment must necessarily hold in more complex environments such as long-horizon agentic workflows. Indeed, the claims can be empirically validated with no LLM-based agent environment at all, since the problem is one of variance and not of language modelling.

Our simulation environment is a chain of $T$ binary decisions. At every step, the logging policy $\pi_l$ plays action 1 with probability $q_l = 0.5$, and a candidate policy $\pi_q$ plays it with probability $q$. Each step yields a Bernoulli reward with mean 0.7 if action 1 was taken and 0.3 otherwise, and the terminal reward of a trajectory is the mean of its step rewards, $r \in [0, 1]$.

The value of any policy in this family is known exactly and is independent of the horizon. For the logging policy, $V(\pi_l) = 0.5(0.7) + 0.5(0.3) = 0.5$, and for a candidate policy $\pi_q$ it is

$$V(\pi_q) = q(0.7) + (1 - q)(0.3) = 0.3 + 0.4q \tag{7}$$

Throughout this section, the candidate policy $\pi_c$ is $\pi_q$ with $q = 0.7$, so $V(\pi_c) = 0.58$.

The environment is deliberately kind to estimators: rewards are bounded, overlap is generous, and the per-step importance weight takes only two values, $0.7/0.5 = 1.4$ when the logged action was 1 and $0.3/0.5 = 0.6$ when it was 0. Whatever fails here must fail in any realistic agent setting.

The per-step weight variance is $\mathrm{Var}(w_t) = 0.16$, and steps are independent by construction, so the illustrative formula of Section 3 is exact here: the trajectory

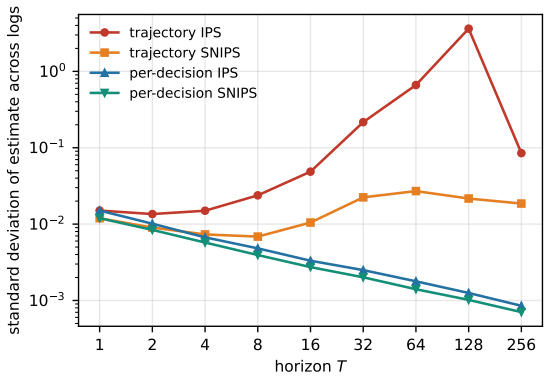


Figure 1: Standard deviation, across 200 independent logs of $n = 2000$ trajectories, of each estimate of $V(\pi_c)$ as a function of the horizon $T$ (both axes logarithmic). The trajectory-level estimators degrade with $T$; the per-decision estimators improve, because a log of fixed size contains $nT$ decision records. The fall in the trajectory-IPS curve past $T = 128$ is degeneration, not recovery; see text.

weight has variance $1.16^T - 1$. The compounding is unremarkable at first—0.16 at $T = 1$, 2.3 at $T = 8$—and then it is not: 115 at $T = 32$, roughly $1.3 \times 10^4$ at $T = 64$, and roughly $3.2 \times 10^{16}$ at $T = 256$. A per-step variance that would be considered benign in a practical contextual bandit setting becomes, two hundred and fifty-six multiplications later, seventeen orders of magnitude higher. In agent settings, on full trajectories, where one decision equals one token, $T$ equals the number of generated tokens—typically far higher.

To observe the consequences, 200 independent logs of $n = 2000$ trajectories were generated under $\pi_l$ for each horizon $T \in \{1, 2, 4, \ldots, 256\}$, and all four estimators of the preceding sections—trajectory-level and per-decision IPS and SNIPS—were computed on each log. Figure 1 shows the standard deviation of each estimate across the 200 logs.

Trajectory-level IPS behaves exactly as the weight variance dictates for as long as the sample can still express that variance: its spread grows from 0.015 at $T = 1$ to 3.6 at $T = 128$, at which point the uncertainty of the estimate is several times larger than the entire range of possible rewards and the estimate carries no information at all. Past that point the failure changes character rather than abating: at $T = 256$ the spread falls back to 0.085, not because the estimator has recovered but because the enormous weights that carry its expectation are no longer sampled at all—the typical trajectory weight has become astronomically small, and 86% of logs return an estimate below 0.01 against a true value of 0.58. The explosion is no longer visible as scatter; it reappears as a silently, confidently wrong answer. The per-decision estimators move in the opposite direction throughout, their spread *shrinking* from 0.012 to 0.0007 over the same sweep, since a log of $n$ trajectories of horizon $T$ contains $nT$ single-factor records

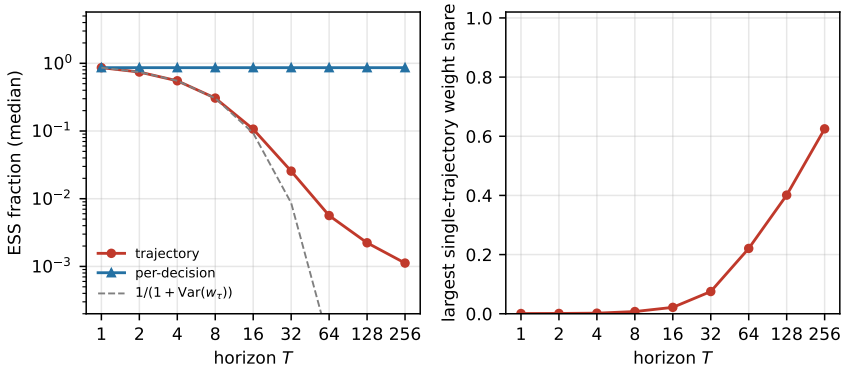


Figure 2: Left: median effective sample size as a fraction of the log, for trajectory-level and per-decision weights, with the prediction $1/(1+\mathrm{Var}(w_\tau))$, $\mathrm{Var}(w_\tau) = 1.16^T - 1$, overlaid; at the largest horizons the prediction falls below what a finite sample can realise and exits the plot. Right: median share of the total trajectory-level weight carried by the single largest weight. By $T = 256$ the trajectory-level log of 2000 trajectories is effectively two trajectories, and one of them carries nearly two thirds of all the weight.

and longer horizons therefore mean more data, not more variance.

Trajectory-level SNIPS appears, based on standard deviations, to survive (Figure 1): its spread never exceeds 0.027 anywhere in the sweep, held down by the boundedness argued in Section 2. The survival is cosmetic; the effective sample size reveals why. For weights $w_1, \ldots, w_n$ the effective sample size $\mathrm{ESS} = (\sum_i w_i)^2 / \sum_i w_i^2$ measures how many equally-weighted observations the weighted log is worth; its expected fraction of $n$ is approximately $1/(1+\mathrm{Var}(w_\tau))$, because with $n$ large the denominator $\frac{1}{n}\sum_i w_i^2$ approaches $\mathbb{E}[w_\tau^2] = 1 + \mathrm{Var}(w_\tau)$ while the numerator $\left(\frac{1}{n}\sum_i w_i\right)^2$ approaches $\mathbb{E}[w_\tau]^2 = 1$.

Figure 2 shows the median ESS fraction collapsing along this predicted curve until the prediction outruns the sample itself: beyond $T \approx 32$ the predicted fraction falls below anything 2000 draws can realise, and the empirical median floors above it, reaching 0.0011 at $T = 256$—of the 2000 logged trajectories, the equivalent of about *two* remain. The right panel shows the same collapse from the other side: the median share of the total weight carried by the single largest trajectory rises from 0.1% to 63%. SNIPS still returns a number in $[0, 1]$, but it is a weighted average computed from a vanishing handful of trajectories, and the cost of that is paid in the next section, where it matters which of several candidates is better. The per-decision ESS fraction is, by contrast, 0.862 at every horizon, unaffected by trajectory length.

# 6 Simulation: A Known, Tolerable Bias

The per-decision estimators escaped the variance explosion, but as discussed, they estimate a biased quantity: each decision is credited with an outcome produced under the logging policy's continuation. The advantage of a simulated environment is that this bias can be computed exactly and then compared against what the estimator actually does.

Recall that the reward attached to every decision record is not the reward of that decision alone but the trajectory's terminal reward, the mean $r = \frac{1}{T}\sum_s r_s$ of all $T$ step rewards. Consider the record for step $t$, which the estimator weights as $w_t\, r$. The weight $w_t$ knows only about the action taken at step $t$, so of the $T$ step rewards bundled inside $r$, it corrects exactly one. For the matched step, the usual IPS argument applies and $\mathbb{E}[w_t\, r_t]$ equals the candidate's per-step value. Every other step reward $r_s$, $s \neq t$, was earned by the logging policy and is independent of $w_t$; since $\mathbb{E}[w_t] = 1$, these contributions pass through the weighting unchanged, still carrying the logging policy's per-step value. Each record is therefore, in expectation, one part candidate value and $T-1$ parts logging value. Pooling over all records preserves this proportion, and the per-decision estimators converge not to $V(\pi_c)$ but to the blend

$$\mathbb{E}\Big[\hat{V}^{\mathrm{dec}}(\pi_c)\Big] \;\longrightarrow\; \frac{1}{T}\,V(\pi_c) + \Big(1-\frac{1}{T}\Big)V(\pi_l), \qquad \text{bias} = \Big(1-\frac{1}{T}\Big)\big(V(\pi_l)-V(\pi_c)\big). \tag{8}$$

For the IPS variant the blend holds exactly in expectation at every $n$; for SNIPS it holds in the limit of a growing log, in keeping with the finite-sample bias discussed in Section 2. Three properties emerge. The bias is *bounded*: its magnitude never exceeds $|V(\pi_c) - V(\pi_l)|$, in contrast to the unbounded variance of Section 5. Its *direction is known*: the estimate is always pulled toward the logging policy's value and never past it, so it understates the candidate's difference from the status quo rather than exaggerating it. And it is *predictable*: in this environment Equation (8) gives the bias in closed form, and the simulation matches it almost exactly—across the full horizon sweep the mean of per-decision SNIPS agrees with the predicted blend to the fourth decimal (at $T = 16$, 0.50499 observed against 0.50500 predicted; Figure 3).

Equation (8) also makes precise *what* the per-decision estimator measures: the value of deviating to the candidate for a single decision and otherwise continuing as the logging policy would have—a one-step deviation value. Crucially, this quantity is affine and strictly increasing in $V(\pi_c)$. Every comparison the estimate exists to inform is therefore preserved in expectation: if one candidate is truly better than another, its per-decision estimate is higher; if a candidate truly improves on the logging policy, its estimate exceeds the logging policy's. What the bias distorts is the *magnitude*, shrinking every difference by the factor $1/T$, not the *ordering*.

Whether the ordering survives on the single finite log one actually has is the practical question, and it is tested directly. Five candidates spanning $q_c \in \{0.30, 0.40, 0.55, 0.65, 0.80\}$ were evaluated on each of 200 logs at the hardest configuration of Section 5, $T = 256$ and $n = 2000$; Figure 4 shows the results.

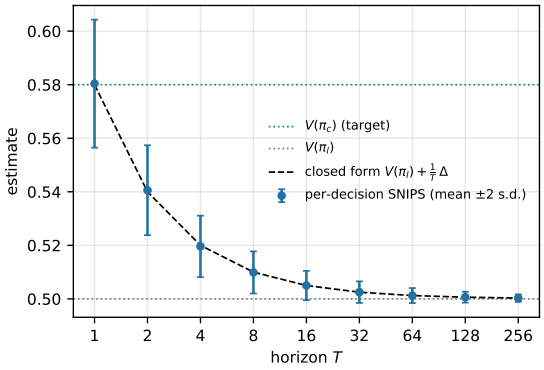


Figure 3: Mean of the per-decision SNIPS estimate across 200 logs (±2 standard deviations) against the closed-form blend of Equation (8), with $\Delta = V(\pi_c) - V(\pi_l)$. The bias is real, but it is exactly the predicted, bounded pull toward $V(\pi_l)$; the estimate never overshoots and never scatters.

Trajectory-level SNIPS is centred near the true values but scatters across them: it recovers the exact ranking of all five candidates in 48.5% of logs, and orders pairs of candidates correctly 91.4% of the time. Per-decision SNIPS produces estimates confined to a band only $1/T$ as wide, yet within that band, the five candidates appear in the correct order in 100% of logs. Every pairwise comparison is correct, and the sign of each candidate's improvement over the logging policy's own (on-policy) estimate is correct in every log for every candidate, including the modest $q_c = 0.55$, whose true improvement is only 0.02. The biased estimator, read as a comparator rather than as a point estimate of $V(\pi_c)$, makes the right call in every instance where the unbiased one sometimes fails.

# 7 Limitations and Conclusion

The method's limits follow from what the per-decision estimator actually estimates. Its target is the one-step deviation value of Section 6: the value of the candidate's choices at the histories the logging policy visited, never in the histories the candidate's own compounded deviations would create. A candidate whose individual decisions look favourable at logged histories, but which would steer trajectories somewhere worse, is over-credited. The estimate is therefore a reliable *comparator* for candidates close to the deployed policy: a revised prompt, an adjusted tool-selection distribution, a retuned temperature. It is no substitute for evaluating a radically different agent, whose relevant histories no logged data contains.

The log and the decisions must also cooperate. Propensities must be recorded at decision time under the sampling distribution actually used, and the logging policy must be genuinely stochastic where it matters: greedy decoding yields

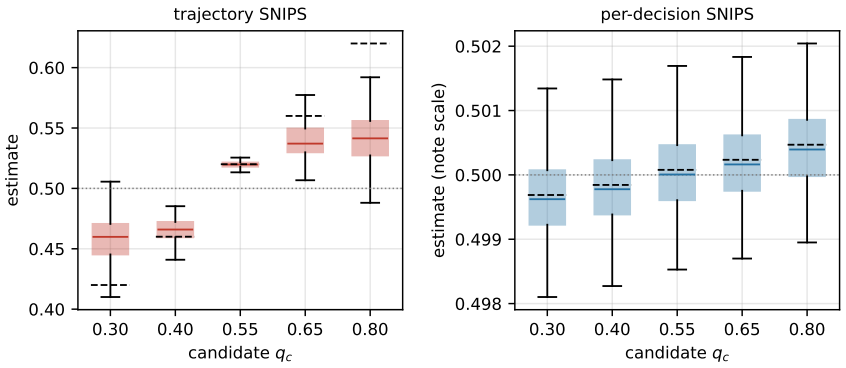


Figure 4: Five candidate policies $q_c \in \{0.30, 0.40, 0.55, 0.65, 0.80\}$ evaluated on the same 200 logs ($T = 256$, $n = 2000$). Boxes span the interquartile range across logs; dashed marks show each estimator's expected value—the true $V(\pi_q)$ on the left, the blend of Equation (8) on the right. Note the different vertical scales: the trajectory-level estimates aim at the truth and miss it noisily, while the per-decision estimates aim at a target $1/T$ as spread out and hit it, in the correct order, every time.

logs that support no reweighting at all, and near-zero temperatures recreate the weight explosion of Section 3 in miniature. Decisions must have tractable action spaces, and the task must be decomposable into decision points whose local quality bears on the outcome, without which the estimate is not wrong so much as uninformative. These are real constraints, but they describe the large and growing class of agentic systems already built as explicit sequences of discrete decisions.

Within this class of problems, the conclusion is simple. Trajectory-level off-policy evaluation of agents is intractable by construction: the weight is a product, its variance compounds exponentially in the horizon, and past the wall the failure does not even announce itself as noise. Per-decision evaluation is tractable at the price of a known bias—bounded, pulled toward the logged behaviour, and, as the simulations show, preserving ordering, making the estimate usable for comparison between candidate agents. Agents decompose their tasks into decisions; their evaluation should decompose the same way.