Contents

1 Chapter 2 — Probability, Measure, and Click Models

François Fenning

1.1 2.1 Motivation: Why Search Needs Measure Theory

The attribution puzzle. Consider a simple question: *What is the probability that a user clicks on the third-ranked product?*

In Chapter 0's toy simulator, we answered this with a lookup table: position 3 gets examination probability 0.7, product quality determines click probability given examination. In Chapter 1, we formalized rewards as expectations over stochastic outcomes ω. But we haven't yet made the **probability space** rigorous.

When the outcome space stops being finite — for example, **continuous** state/features (user embeddings $u \in \mathbb{R}^d$, product features $p \in \mathbb{R}^f$) or **infinite-horizon trajectories** in an RL formulation $(S_0, A_0, R_0, S_1, A_1, R_1, \ldots)$ — the "probability = number of favourable outcomes ÷ number of possible outcomes" story breaks down. Naive counting no longer works; we need:

1. **Measure-theoretic probability** on general spaces
2. **Lebesgue integrals / expectations** to define values and policy gradients
3. **Product σ-algebras** to talk about probabilities on trajectories, stopping times, etc.
4. **Radon–Nikodym derivatives** for importance sampling and off-policy evaluation

The click model problem. Search systems must answer: *Given a ranking $\pi = (p_1, \ldots, p_M)$, what is the distribution over click patterns $C \subseteq \{1, \ldots, M\}$?*

Simple models like "top result gets 50% of clicks" are empirically false. Real click behavior exhibits: - **Position bias**: Items ranked higher are examined more often, independent of quality - **Cascade abandonment**: Users scan top-to-bottom, stopping when they find a satisfactory result or lose patience - **Contextual heterogeneity**: Premium users have different click propensities than price hunters

The **Position Bias Model (PBM)** and **Dynamic Bayesian Network (DBN)** formalize these patterns using probability theory on discrete outcome spaces. But to **prove** properties (unbiasedness of estimators, convergence of learning algorithms), we need measure-theoretic foundations.

Chapter roadmap. This chapter builds the probability machinery for RL in continuous spaces:

- **Section 2.2–2.3**: Probability spaces, random variables, conditional expectation (Bourbaki-Kolmogorov rigorous treatment)
- **Section 2.4**: Filtrations and stopping times (for abandonment modeling)
- **Section 2.5**: Position Bias Model (PBM) and Dynamic Bayesian Networks (DBN) for clicks
- **Section 2.6**: Propensity scoring and unbiased estimation (foundation for off-policy learning)

- **Section 2.7**: Computational verification (NumPy experiments)
- **Section 2.8**: RL bridges (MDPs, policy evaluation, OPE preview)

Why this matters for RL. Chapters 1 wrote $\mathbb{E}[R \mid \mathbf{w}]$ informally. Now we make it precise: expectations are **Lebesgue integrals** over probability measures. Policy gradients (Chapter 8) require interchanging ∇_θ with $\mathbb{E}$—justified by Dominated Convergence. Off-policy evaluation (Chapter 9) uses importance sampling—defined via Radon-Nikodym derivatives. Without this chapter's foundations, those algorithms are heuristics. With them, they are theorems.

Let's begin.

1.2 2.2 Probability Spaces and Random Variables

We start with Kolmogorov's axiomatization of probability (1933), the foundation for modern stochastic processes and reinforcement learning.

1.2.1 2.2.1 Measurable Spaces and σ-Algebras

Definition 2.2.1 (Measurable Space) {#DEF-2.2.1}

A **measurable space** is a pair $(\Omega, \mathcal{F})$ where: 1. Ω is a nonempty set (the **sample space**) 2. $\mathcal{F}$ is a σ-algebra on Ω: a collection of subsets of Ω satisfying: - $\Omega \in \mathcal{F}$ - If $A \in \mathcal{F}$, then $A^c := \Omega \setminus A \in \mathcal{F}$ (closed under complements) - If $A_1, A_2, ... \in \mathcal{F}$, then $\bigcup_{n=1}^{\infty} A_n \in \mathcal{F}$ (closed under countable unions)

Elements of $\mathcal{F}$ are called **measurable sets** or **events**.

Example 2.2.1 (Finite outcome spaces). If $\Omega = \{\omega_1, \dots, \omega_N\}$ is finite, the **power set** $\mathcal{F} = 2^\Omega$ (all subsets) is a σ-algebra. This suffices for tabular RL and discrete click models.

Example 2.2.2 (Borel σ-algebra on $\mathbb{R}$). Let $\Omega = \mathbb{R}$. The **Borel σ-algebra** $\mathcal{B}(\mathbb{R})$ is the smallest σ-algebra containing all open intervals (a, b). This enables probability on continuous spaces (e.g., user embeddings, boost weights).

Remark 2.2.1 (Why σ-algebras?). Why not allow *all* subsets as events? Two reasons:

1. **Pathological sets exist**: On $\mathbb{R}$, non-measurable sets (Vitali's construction) would violate additivity axioms if assigned probability
2. **Functional analysis**: Measurable functions (next) form well-behaved vector spaces; arbitrary functions do not

The σ-algebra structure ensures probability theory is **consistent** (no contradictions) and **complete** (all natural events are measurable).

1.2.2 2.2.2 Probability Measures

Definition 2.2.2 (Probability Measure) {#DEF-2.2.2}

A **probability measure** on $(\Omega, \mathcal{F})$ is a function $\mathbb{P} : \mathcal{F} \to [0, 1]$ satisfying: 1. **Normalization**: $\mathbb{P}(\Omega) = 1$ 2. **Non-negativity**: $\mathbb{P}(A) \geq 0$ for all $A \in \mathcal{F}$ 3. **Countable additivity** (σ-additivity): For any countable sequence of **disjoint** events $A_1, A_2, ... \in \mathcal{F}$ (i.e., $A_i \cap A_j = \emptyset$ for $i \neq j$),

$$\mathbb{P}\left(\bigcup_{n=1}^{\infty} A_n\right) = \sum_{n=1}^{\infty} \mathbb{P}(A_n).$$

The triple $(\Omega, \mathcal{F}, \mathbb{P})$ is called a **probability space**.

Example 2.2.3 (Discrete uniform distribution). Let $\Omega = \{1, 2, \ldots, N\}$, $\mathcal{F} = 2^\Omega$. Define $\mathbb{P}(A) = |A|/N$ for all $A \subseteq \Omega$. This is a probability measure (verify: normalization holds, countable additivity reduces to finite additivity since Ω is finite).

Example 2.2.4 (Uniform distribution on $[0,1]$). Let $\Omega = [0,1]$, $\mathcal{F} = \mathcal{B}([0,1])$ (Borel sets). Define $\mathbb{P}((a,b)) = b - a$ for intervals $(a,b) \subseteq [0,1]$. This extends uniquely to all Borel sets by **Carathéodory's Extension Theorem** [@folland:real_analysis:1999, Theorem 1.14], giving the **Lebesgue measure** restricted to $[0,1]$.

Remark 2.2.2 (Necessity of countable additivity). Why require *countable* additivity rather than just finite additivity? Consider the Lebesgue measure of singletons in $[0,1]$: if $\mathbb{P}(\{x\}) > 0$ for all $x \in [0,1]$, then σ-additivity forces

$$\mathbb{P}([0,1]) = \sum_{x \in [0,1]} \mathbb{P}(\{x\}) = \infty,$$

contradicting normalization. Only σ-additivity eliminates such inconsistencies on uncountable spaces. Finite additivity is too weak for continuous probability.

1.2.3 2.2.3 Random Variables

Definition 2.2.3 (Random Variable) {#DEF-2.2.3}

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and $(E, \mathcal{E})$ a measurable space. A function $X : \Omega \to E$ is a **random variable** if it is $(\mathcal{F}, \mathcal{E})$**-measurable**: for all $A \in \mathcal{E}$,

$$X^{-1}(A) := \{\omega \in \Omega : X(\omega) \in A\} \in \mathcal{F}.$$

Intuition: Pre-images of measurable sets are measurable. This ensures $\mathbb{P}(X \in A)$ is well-defined for all events $A \in \mathcal{E}$.

Example 2.2.5 (Click indicator). In a search session, let Ω represent all possible user behaviors (examination patterns, clicks, purchases). Define $X_k : \Omega \to \{0,1\}$ by $X_k(\omega) = 1$ if user clicks on result k under outcome ω, and $X_k(\omega) = 0$ otherwise. Then X_k is a random variable (discrete codomain).

Example 2.2.6 (GMV as a random variable). Let Ω be the space of all search sessions (rankings, clicks, purchases). Define $\text{GMV} : \Omega \to \mathbb{R}_+$ by summing purchase prices. Then GMV is a non-negative real-valued random variable.

Proposition 2.2.1 (Measurability of compositions) {#THM-2.2.1}. If $X : \Omega_1 \to \Omega_2$ is $(\mathcal{F}_1, \mathcal{F}_2)$-measurable and $f : \Omega_2 \to \Omega_3$ is $(\mathcal{F}_2, \mathcal{F}_3)$-measurable, then $f \circ X : \Omega_1 \to \Omega_3$ is $(\mathcal{F}_1, \mathcal{F}_3)$-measurable.

Proof. For $A \in \mathcal{F}_3$,

$$(f \circ X)^{-1}(A) = X^{-1}(f^{-1}(A)).$$

Since f is measurable, $f^{-1}(A) \in \mathcal{F}_2$. Since X is measurable, $X^{-1}(f^{-1}(A)) \in \mathcal{F}_1$. $\square$

Remark 2.2.2 (Inverse-image composition technique). The proof uses the inverse-image composition identity $(f \circ X)^{-1}(A) = X^{-1}(f^{-1}(A))$ and closure of σ-algebras under inverse images. This "inverse-image trick" will reappear when showing measurability of stopped processes in Section 2.4.

Remark 2.2.3 (RL preview). In RL, states S_t, actions A_t, rewards R_t are all random variables on a common probability space $(\Omega, \mathcal{F}, \mathbb{P})$ induced by the policy π and environment dynamics. Measurability ensures $\mathbb{P}(R_t > r)$ is well-defined for all thresholds r.

1.2.4 2.2.4 Expectation and Integration

Definition 2.2.4 (Expectation) {#DEF-2.2.4}

Let $X : \Omega \to \mathbb{R}$ be a random variable on $(\Omega, \mathcal{F}, \mathbb{P})$. The **expectation** (or **expected value**) of X is

$$\mathbb{E}[X] := \int_\Omega X\, d\mathbb{P},$$

where the integral is the **Lebesgue integral** with respect to the probability measure $\mathbb{P}$. We say X is **integrable** if $\mathbb{E}[|X|] < \infty$.

Construction (standard three-step approach, from [@folland:real_analysis:1999, Chapter 2]): 1. **Simple functions**: For $s = \sum_{i=1}^n a_i \mathbf{1}_{A_i}$ with $A_i \in \mathcal{F}$ disjoint,

$$\int_\Omega s\, d\mathbb{P} := \sum_{i=1}^n a_i \mathbb{P}(A_i).$$

2. **Non-negative functions**: For $X \geq 0$, approximate by simple functions $s_n \uparrow X$:

$$\int_\Omega X\, d\mathbb{P} := \sup_n \int_\Omega s_n\, d\mathbb{P}.$$

3. **General functions**: Decompose $X = X^+ - X^-$ where $X^+ = \max(X, 0)$, $X^- = \max(-X, 0)$:

$$\int_\Omega X\, d\mathbb{P} := \int_\Omega X^+\, d\mathbb{P} - \int_\Omega X^-\, d\mathbb{P}$$

provided both integrals are finite.

Example 2.2.7 (Finite sample space). Let $\Omega = \{\omega_1, \dots, \omega_N\}$ with $\mathbb{P}(\{\omega_i\}) = p_i$. Then

$$\mathbb{E}[X] = \sum_{i=1}^N X(\omega_i) p_i.$$

This is the familiar discrete expectation formula.

Example 2.2.8 (Continuous uniform on $[0, 1]$). Let $X(\omega) = \omega$ for $\omega \in [0, 1]$ with Lebesgue measure. Then

$$\mathbb{E}[X] = \int_0^1 x\, dx = \frac{1}{2}.$$

Theorem 2.2.2 (Linearity of Expectation) {#THM-2.2.2}

If X, Y are integrable random variables and $\alpha, \beta \in \mathbb{R}$, then $\alpha X + \beta Y$ is integrable and

$$\mathbb{E}[\alpha X + \beta Y] = \alpha \mathbb{E}[X] + \beta \mathbb{E}[Y].$$

Proof. This follows from linearity of the Lebesgue integral [@folland:real_analysis:1999, Proposition 2.12]. □

Remark 2.2.5 (Linearity via simple-function approximation). The mechanism is the linearity of the Lebesgue integral, proved by reducing non-negative functions to increasing simple-function approximations and extending to integrable functions via $X = X^+ - X^-$. Naming the technique clarifies that no independence assumptions are needed—linearity is purely measure-theoretic.

Theorem 2.2.3 (Monotone Convergence Theorem) {#THM-2.2.3}

Let $0 \leq X_1 \leq X_2 \leq \cdots$ be a non-decreasing sequence of non-negative random variables with $X_n \to X$ pointwise. Then

$$\mathbb{E}[X] = \lim_{n \to \infty} \mathbb{E}[X_n].$$

Proof. Direct application of the Monotone Convergence Theorem for Lebesgue integration [@folland:real_analysis:1999, Theorem 2.14]. □

Remark 2.2.6 (Monotone convergence technique). The key mechanism is monotone convergence: approximate X by an increasing sequence $X_n \uparrow X$ of simple functions and pass the limit inside the integral. No domination is required; monotonicity alone suffices.

Remark 2.2.4 (RL preview: reward expectations). In RL, the value function $V^\pi(s) = \mathbb{E}^\pi[\sum_{t=0}^\infty \gamma^t R_t \mid S_0 = s]$ is an expectation over trajectories. For this to be well-defined, we need R_t to be measurable and integrable. The Monotone Convergence Theorem (THM-2.2.3) allows us to interchange limits and expectations when computing Bellman operator fixed points (Chapter 3).

1.3 2.3 Conditional Probability and Conditional Expectation

Click models require **conditional probabilities**: the probability of clicking given examination, the probability of examination given position. We formalize this rigorously.

1.3.1 2.3.1 Conditional Probability Given an Event

Definition 2.3.1 (Conditional Probability) {#DEF-2.3.1}

Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and $B \in \mathcal{F}$ with $\mathbb{P}(B) > 0$. For any event $A \in \mathcal{F}$, the **conditional probability** of A given B is

$$\mathbb{P}(A \mid B) := \frac{\mathbb{P}(A \cap B)}{\mathbb{P}(B)}.$$

Theorem 2.3.1 (Law of Total Probability) {#THM-2.3.1}

Let $B_1, B_2, \ldots \in \mathcal{F}$ be a countable partition of Ω (disjoint events with $\bigcup_n B_n = \Omega$) such that $\mathbb{P}(B_n) > 0$ for all n. Then for any event $A \in \mathcal{F}$,

$$\mathbb{P}(A) = \sum_{n=1}^{\infty} \mathbb{P}(A \mid B_n)\mathbb{P}(B_n).$$

Proof.

Step 1 (Partition property): Since $\{B_n\}$ partition Ω and are disjoint,

$$A = A \cap \Omega = A \cap \left(\bigcup_{n=1}^{\infty} B_n\right) = \bigcup_{n=1}^{\infty} (A \cap B_n),$$

with the sets $A \cap B_n$ pairwise disjoint.

Step 2 (Apply σ-additivity): By countable additivity of $\mathbb{P}$,

$$\mathbb{P}(A) = \mathbb{P}\left(\bigcup_{n=1}^{\infty} (A \cap B_n)\right) = \sum_{n=1}^{\infty} \mathbb{P}(A \cap B_n).$$

Step 3 (Substitute definition of conditional probability): By Definition 2.3.1, $\mathbb{P}(A \cap B_n) = \mathbb{P}(A \mid B_n)\mathbb{P}(B_n)$. Substituting:

$$\mathbb{P}(A) = \sum_{n=1}^{\infty} \mathbb{P}(A \mid B_n)\mathbb{P}(B_n).$$

□

Remark 2.3.1 (The partition technique). This proof uses the **partition technique**: decompose a complex event into disjoint cases, apply additivity, and sum. We'll use this repeatedly when analyzing click cascades (Section 2.5).

Example 2.3.1 (Click given examination). In a search session, let $E_k = \{\text{user examines result } k\}$ and $C_k = \{\text{user clicks result } k\}$. The **examination-conditioned click probability** is

$$\mathbb{P}(C_k \mid E_k) = \frac{\mathbb{P}(C_k \cap E_k)}{\mathbb{P}(E_k)}.$$

This is the foundation of the Position Bias Model (PBM, Section 2.5).

1.3.2 2.3.2 Conditional Expectation Given a σ-Algebra

For RL applications (policy evaluation, off-policy estimation), we need conditional expectation **with respect to a σ-algebra**, not just a single event. This is more abstract but essential.

Definition 2.3.2 (Conditional Expectation Given σ-Algebra) {#DEF-2.3.2}

Let X be an integrable random variable on $(\Omega, \mathcal{F}, \mathbb{P})$ and $\mathcal{G} \subseteq \mathcal{F}$ a sub-σ-algebra. The **conditional expectation** of X given $\mathcal{G}$, denoted $\mathbb{E}[X \mid \mathcal{G}]$, is the unique (up to $\mathbb{P}$-almost everywhere equality) $\mathcal{G}$-measurable random variable Y satisfying:

1. **Measurability**: Y is $\mathcal{G}$-measurable
2. **Partial averaging**: For all $A \in \mathcal{G}$,

$$\int_A Y \, d\mathbb{P} = \int_A X \, d\mathbb{P}.$$

Intuition: $\mathbb{E}[X \mid \mathcal{G}]$ is the "best $\mathcal{G}$-measurable approximation" to X. It averages X over the "unobservable" parts not captured by $\mathcal{G}$.

Example 2.3.2 (Trivial cases). - If $\mathcal{G} = \{\emptyset, \Omega\}$ (trivial σ-algebra), then $\mathbb{E}[X \mid \mathcal{G}] = \mathbb{E}[X]$ (constant function). - If $\mathcal{G} = \mathcal{F}$ (full σ-algebra), then $\mathbb{E}[X \mid \mathcal{G}] = X$ (no averaging).

Theorem 2.3.2 (Tower Property) {#THM-2.3.2}

Let $\mathcal{G} \subseteq \mathcal{H} \subseteq \mathcal{F}$ be nested σ-algebras. Then

$$\mathbb{E}[\mathbb{E}[X \mid \mathcal{H}] \mid \mathcal{G}] = \mathbb{E}[X \mid \mathcal{G}].$$

Proof. Let $Y = \mathbb{E}[X \mid \mathcal{H}]$ and $Z = \mathbb{E}[X \mid \mathcal{G}]$. For any $A \in \mathcal{G}$, since $\mathcal{G} \subseteq \mathcal{H}$ we have

$$\int_A Y \, d\mathbb{P} = \int_A X \, d\mathbb{P} = \int_A Z \, d\mathbb{P}.$$

Define $W := \mathbb{E}[Y \mid \mathcal{G}]$. By the defining property of conditional expectation, W is the unique $\mathcal{G}$-measurable random variable such that $\int_A W \, d\mathbb{P} = \int_A Y \, d\mathbb{P}$ for all $A \in \mathcal{G}$. Since Z also satisfies $\int_A Z \, d\mathbb{P} = \int_A Y \, d\mathbb{P}$ for all $A \in \mathcal{G}$, uniqueness implies $W = Z$ almost surely. Hence $\mathbb{E}[\mathbb{E}[X \mid \mathcal{H}] \mid \mathcal{G}] = \mathbb{E}[X \mid \mathcal{G}]$. □

Remark 2.3.3 (Tower as projection/uniqueness). The technique is the projection/uniqueness property of conditional expectation: $\mathbb{E}[\cdot \mid \mathcal{G}]$ is the L^1 projection onto $\mathcal{G}$-measurable functions characterized by matching integrals on sets in $\mathcal{G}$. This viewpoint will reappear in martingale proofs.

Theorem 2.3.3 (Existence and Uniqueness of Conditional Expectation) {#THM-2.3.3}

Let $X \in L^1(\Omega, \mathcal{F}, \mathbb{P})$ and $\mathcal{G} \subseteq \mathcal{F}$ a sub-σ-algebra. Then $\mathbb{E}[X \mid \mathcal{G}]$ exists and is unique up to $\mathbb{P}$-almost sure equality.

Proof. This is a deep result from measure theory, proven via the **Radon-Nikodym Theorem** [@folland:real_analysis:1999, Theorem 3.8]. We cite this result and defer the full proof to standard references. The key idea: define a signed measure $\nu(A) = \int_A X \, d\mathbb{P}$ for $A \in \mathcal{G}$. This measure is absolutely continuous with respect to $\mathbb{P}$ restricted to $\mathcal{G}$. The Radon-Nikodym Theorem provides the density $d\nu/d\mathbb{P}$, which is precisely $\mathbb{E}[X \mid \mathcal{G}]$. □

Remark 2.3.2 (The Radon-Nikodym connection). Conditional expectation is fundamentally a **change of measure** problem. This reappears in off-policy RL (Chapter 9): importance sampling ratios $\rho_t = \pi(a_t \mid s_t)/\mu(a_t \mid s_t)$ are Radon-Nikodym derivatives relating two policies.

1.4 2.4 Filtrations and Stopping Times

Session abandonment in search is a **sequential stopping problem**: users scan results top-to-bottom, stopping when satisfied or losing patience. Formalizing this requires **filtrations** and **stopping times**.

1.4.1 2.4.1 Filtrations

Definition 2.4.1 (Filtration) {#DEF-2.4.1}

A **filtration** on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ is a sequence of σ-algebras $\{\mathcal{F}_t\}_{t=0}^{\infty}$ satisfying:

$$\mathcal{F}_0 \subseteq \mathcal{F}_1 \subseteq \mathcal{F}_2 \subseteq \cdots \subseteq \mathcal{F}.$$

Intuition: $\mathcal{F}_t$ represents "information available up to time t". As t increases, more information is revealed.

Example 2.4.1 (Search session filtration). In a search session with M results, let $\mathcal{F}_k$ be the σ-algebra generated by examination and click outcomes for positions $1, \dots, k$:

$$\mathcal{F}_k = \sigma(E_1, C_1, E_2, C_2, \dots, E_k, C_k).$$

At stage k, the user has seen results 1 through k; outcomes at positions $k+1, \dots, M$ are not yet revealed.

Definition 2.4.2 (Adapted Process) {#DEF-2.4.2}

A sequence of random variables $\{X_t\}_{t=0}^{\infty}$ is **adapted** to filtration $\{\mathcal{F}_t\}$ if X_t is $\mathcal{F}_t$-measurable for all t.

Intuition: X_t depends only on information available up to time t (no "looking into the future").

1.4.2 2.4.2 Stopping Times

Definition 2.4.3 (Stopping Time) {#DEF-2.4.3}

Let $\{\mathcal{F}_t\}$ be a filtration on $(\Omega, \mathcal{F}, \mathbb{P})$. A random variable $\tau : \Omega \to \mathbb{N} \cup \{\infty\}$ is a **stopping time** if for all $t \in \mathbb{N}$,

$$\{\tau = t\} \in \mathcal{F}_t.$$

Intuition: The event "we stop at time t" is determined by information available **up to and including** time t. No future information is used to decide when to stop.

Example 2.4.2 (First click is a stopping time). Define

$$\tau = \min\{k \geq 1 : C_k = 1\},$$

the first position where the user clicks (or $\tau = \infty$ if no clicks). Then τ is a stopping time: $\{\tau = k\} = \{C_1 = 0, \dots, C_{k-1} = 0, C_k = 1\} \in \mathcal{F}_k$.

Example 2.4.3 (Abandonment stopping time). Model session abandonment as

$$\tau = \min\{k \geq 1 : E_k = 0\},$$

the first position the user does not examine (or $\tau = \infty$ if user examines all M results). This is a stopping time: $\{\tau = k\} = \{E_1 = 1, \dots, E_{k-1} = 1, E_k = 0\} \in \mathcal{F}_k$.

Non-Example 2.4.1 (Last click is NOT a stopping time). Define $\tau = \max\{k : C_k = 1\}$, the position of the last click. This is **not** a stopping time: to know $\{\tau = k\}$, we must verify $C_{k+1} = 0, \dots, C_M = 0$, requiring future information beyond time k.

Theorem 2.4.1 (Measurability at a Stopping Time) {#THM-2.4.1}

If $\{X_t\}$ is adapted to $\{\mathcal{F}_t\}$ and τ is a stopping time, then X_τ (defined as $X_\tau(\omega) = X_{\tau(\omega)}(\omega)$ when $\tau(\omega) < \infty$) is measurable with respect to $\mathcal{F}_\tau := \{A \in \mathcal{F} : A \cap \{\tau = t\} \in \mathcal{F}_t \text{ for all } t\}$.

Proof.

Step 1 (Reduce to cylinder events). It suffices to show $\{X_\tau \in B\} \in \mathcal{F}_\tau$ for all Borel $B \subseteq \mathbb{R}$, since X_τ is real-valued.

Step 2 (Slice by deterministic times). For each $t \in \mathbb{N}$, define $A_t := \{\tau = t\} \cap \{X_t \in B\}$. Then

$$\{X_\tau \in B\} = \bigcup_{t=0}^{\infty} A_t,$$

because on $\{\tau = t\}$ we have $X_\tau = X_t$.

Step 3 (Measurability of slices). Since τ is a stopping time, $\{\tau = t\} \in \mathcal{F}_t$. Adaptation implies X_t is $\mathcal{F}_t$-measurable, so $\{X_t \in B\} \in \mathcal{F}_t$. Hence $A_t \in \mathcal{F}_t$ for all t.

Step 4 (Definition of $\mathcal{F}_\tau$). By definition, $E \in \mathcal{F}_\tau$ iff $E \cap \{\tau = t\} \in \mathcal{F}_t$ for all t. For $E = \{X_\tau \in B\}$ and each t:

$$E \cap \{\tau = t\} = \{\tau = t\} \cap \{X_t \in B\} = A_t \in \mathcal{F}_t.$$

Therefore $E \in \mathcal{F}_\tau$.

Step 5 (Conclusion). Since pre-images of Borel sets under X_τ lie in $\mathcal{F}_\tau$, X_τ is $\mathcal{F}_\tau$-measurable. □

Remark 2.4.2 (Stopping-time measurability technique). The method slices $\{X_\tau \in B\}$ along deterministic times and uses adaptation/stopping-time properties to establish measurability on each slice. This is the **inverse-image + partition** technique, mirroring Remark 2.2.2 and Remark 2.3.1.

Remark 2.4.1 (RL preview: episodic termination). In RL, episode length is often a stopping time: $\tau = \min\{t : \text{terminal state reached}\}$. The return $G = \sum_{t=0}^{\tau} \gamma^t R_t$ is $\mathcal{F}_\tau$-measurable. For infinite-horizon discounted settings, we need $\tau = \infty$ with probability 1 (continuing tasks).

1.5 2.5 Click Models for Search

We now apply probability theory to model **click behavior** in ranked search. The Position Bias Model (PBM) and Dynamic Bayesian Network (DBN) are foundational for search evaluation and off-policy learning.

1.5.1 2.5.1 The Position Bias Model (PBM)

Motivation. Empirical observation: top-ranked results receive disproportionately more clicks, **even when relevance is controlled**. A product ranked at position 1 gets 30% CTR; the same product at position 5 gets 8% CTR. This is **position bias**: users are more likely to examine top positions, independent of content quality.

Definition 2.5.1 (Position Bias Model) {#DEF-2.5.1}

Let $\pi = (p_1, \dots, p_M)$ be a ranking of M products. For each position $k \in \{1, \dots, M\}$, define: - $E_k \in \{0, 1\}$: User examines result at position k (1 = examine, 0 = skip) - $C_k \in \{0, 1\}$: User clicks on result at position k (1 = click, 0 = no click) - $\text{rel}(p_k)$: Relevance (or attractiveness) of product p_k at position k

The **Position Bias Model (PBM)** assumes:

1. **Examination is position-dependent only**:

$$\mathbb{P}(E_k = 1) = \theta_k,$$

where $\theta_k \in [0, 1]$ is the **examination probability** at position k, independent of the product.

2. **Click requires examination and relevance**:

$$C_k = E_k \cdot \text{Bernoulli}(\text{rel}(p_k)),$$

i.e.,

$$\mathbb{P}(C_k = 1 \mid E_k = 1) = \text{rel}(p_k), \quad \mathbb{P}(C_k = 1 \mid E_k = 0) = 0.$$

3. **Independence across positions**: Conditioned on the ranking π, the events $(E_1, C_1), (E_2, C_2), \ldots$ are independent.

Click probability formula:

$$\mathbb{P}(C_k = 1) = \mathbb{P}(C_k = 1 \mid E_k = 1)\mathbb{P}(E_k = 1) = \text{rel}(p_k) \cdot \theta_k. \tag{2.1}$$

{#EQ-2.1}

Example 2.5.1 (PBM parametrization). Suppose examination probabilities decay exponentially with position:

$$\theta_k = \theta_1 \cdot e^{-\lambda(k-1)}, \quad \lambda > 0.$$

For $\theta_1 = 0.9$ and $\lambda = 0.3$: - Position 1: $\theta_1 = 0.90$ - Position 2: $\theta_2 = 0.67$ - Position 3: $\theta_3 = 0.50$ - Position 5: $\theta_5 = 0.27$

A product with $\text{rel}(p) = 0.5$ gets: - At position 1: $\mathbb{P}(C_1 = 1) = 0.5 \times 0.90 = 0.45$ - At position 5: $\mathbb{P}(C_5 = 1) = 0.5 \times 0.27 = 0.135$

Same product, $3\times$ difference in CTR due to position bias alone.

Remark 2.5.1 (Why PBM?). The independence assumption (3) is **empirically false**: users often stop after finding a satisfactory result, inducing **negative dependence** across positions. Despite this, PBM is analytically tractable and a good first-order model. The DBN model (next) relaxes independence. Note: independence is **not** required for the single-position marginal #EQ-2.1; it becomes relevant for multi-position events.

1.5.2 2.5.2 The Dynamic Bayesian Network (DBN) Model

The **cascade hypothesis** [@craswell:cascade:2008]: users scan top-to-bottom, clicking on attractive results, and **stopping** after a satisfactory click. This induces dependence: if position 2 is clicked and satisfies the user, positions 3–M are never examined.

Definition 2.5.2 (Dynamic Bayesian Network Model for Clicks) {#DEF-2.5.2}

For each position $k \in \{1, \ldots, M\}$, define: - $E_k \in \{0, 1\}$: Examination at position k - $C_k \in \{0, 1\}$: Click at position k - $S_k \in \{0, 1\}$: User is satisfied after examining position k (1 = satisfied, stops; 0 = continues)

Convention: S_k is defined only when $E_k = 1$; by convention set $S_k = 0$ when $E_k = 0$ so the cascade is well-defined.

The **DBN cascade model** specifies:

1. **Examination cascade**:

$$\mathbb{P}(E_1 = 1) = 1 \quad \text{(user always examines first result)},$$

$$\mathbb{P}(E_{k+1} = 1 \mid E_k = 1, S_k = 0) = 1, \quad \mathbb{P}(E_{k+1} = 1 \mid \text{otherwise}) = 0. \tag{2.2}$$

{#EQ-2.2}

Intuition: User examines next position if current position was examined but user is not satisfied.

2. **Click given examination**:

$$\mathbb{P}(C_k = 1 \mid E_k = 1) = \mathrm{rel}(p_k), \quad \mathbb{P}(C_k = 1 \mid E_k = 0) = 0.$$

3. **Satisfaction given click**:

$$\mathbb{P}(S_k = 1 \mid C_k = 1) = s(p_k), \quad \mathbb{P}(S_k = 1 \mid C_k = 0) = 0,$$

where $s(p_k) \in [0, 1]$ is the **satisfaction probability** for product p_k.

4. **Abandonment**: Define stopping time

$$\tau = \min\{k : S_k = 1 \text{ or } k = M\},$$

the first position where user is satisfied (or end of list).

Key difference from PBM: Examination at position $k + 1$ depends on outcomes at position k (via S_k). This is a **Markov chain** over positions, not independent Bernoullis.

Proposition 2.5.1 (Marginal examination probability in DBN) {#THM-2.5.1}. Under the DBN model, the probability of examining position k is

$$\mathbb{P}(E_k = 1) = \prod_{j=1}^{k-1} \left[1 - \mathrm{rel}(p_j) \cdot s(p_j)\right]. \tag{2.3}$$

{#EQ-2.3}

Proof.

Step 1 (Base case): $\mathbb{P}(E_1 = 1) = 1$ by model definition. The formula gives $\prod_{j=1}^{0}[\cdots] = 1$ (empty product), so $k = 1$ holds.

Step 2 (Recursive structure): User examines position k if and only if they examined all positions $1, \ldots, k-1$ without being satisfied. By #EQ-2.2, $E_k = 1$ iff $E_{k-1} = 1$ and $S_{k-1} = 0$.

Step 3 (Probability of not being satisfied): At position j, user is satisfied iff $C_j = 1$ and $S_j = 1$. Thus,

$$\mathbb{P}(S_j = 1) = \mathbb{P}(C_j = 1) \cdot s(p_j) = \mathrm{rel}(p_j) \cdot s(p_j).$$

So $\mathbb{P}(S_j = 0) = 1 - \mathrm{rel}(p_j) \cdot s(p_j)$.

Step 4 (Chain rule): Since user must avoid satisfaction at all $j < k$,

$$\mathbb{P}(E_k = 1) = \mathbb{P}(E_1 = 1) \cdot \prod_{j=1}^{k-1} \mathbb{P}(S_j = 0) = \prod_{j=1}^{k-1} \left[1 - \mathrm{rel}(p_j) \cdot s(p_j)\right].$$

□

Remark 2.5.2 (Examination decay in DBN). By #EQ-2.3, examination probability **decays multiplicatively** with position. Each unsatisfactory result provides another chance to abandon. If all products have $\mathrm{rel}(p) \cdot s(p) = 0.2$, then: - $\mathbb{P}(E_1 = 1) = 1.0$ - $\mathbb{P}(E_2 = 1) = 0.8$ - $\mathbb{P}(E_3 = 1) = 0.64$ - $\mathbb{P}(E_5 = 1) = 0.41$

This is empirically more accurate than PBM's position-only dependence.

Remark 2.5.3 (Stopping time interpretation). The abandonment position $\tau = \min\{k : S_k = 1\}$ is a stopping time with respect to the filtration $\mathcal{F}_k = \sigma(E_1, C_1, S_1, \ldots, E_k, C_k, S_k)$. This connects to Section 2.4: user behavior is a stopped random process.

1.5.3 2.5.3 Comparing PBM and DBN

Trade-offs:

Property	PBM	DBN (Cascade)
Independence	Yes (positions independent)	No (cascade dependence)
Realism	Low (ignores abandonment)	High (models stopping)
Analytic tractability	High (closed-form CTR)	Medium (requires recursion)
Parameter estimation	Easy (linear regression)	Harder (EM algorithm)

When to use PBM: Offline analysis, A/B test design, approximate CTR modeling. Fast, simple, interpretable.

When to use DBN: Off-policy evaluation, counterfactual ranking, realistic simulation. More accurate but computationally expensive.

Chapter 0 connection: The toy simulator in Chapter 0 used a simplified PBM (fixed examination probabilities θ_k, independent clicks). Chapter 4's `zoosim` will implement both PBM and DBN, configurable via `zoosim/config.py`.

!!! note "Code ↔ Config (position bias and satisfaction)" PBM and DBN parameters map to configuration fields: - Examination bias vectors: `zoosim/config.py:178` (`BehaviorConfig.pos_bias`) - Satisfaction dynamics weights: `zoosim/config.py:165` (`BehaviorConfig`) KG: `MOD-zoosim.behavior`, `CN-ClickModel`.

!!! note "Code ↔ Behavior (click cascade and termination)" Theory ↔ implementation links: - Position bias lookup: `zoosim/behavior.py:49` (`_position_bias`) - Click + satisfaction update: `zoosim/behavior.py:78` - Purchase and termination: `zoosim/behavior.py:88` These implement PBM/DBN-style examination, click, satisfaction, and stopping. KG: `MOD-zoosim.behavior`, `CN-ClickModel`.

1.6 2.6 Unbiased Estimation via Propensity Scoring

A central challenge in RL for search: we observe clicks under a **logging policy** (current production ranking), but want to evaluate a **new policy** (candidate ranking) without deploying it. This requires **off-policy evaluation (OPE)** via **propensity scoring**.

1.6.1 2.6.1 The Counterfactual Evaluation Problem

Setup: - Logging policy π_0 generates ranking $\pi_0(x)$ for context x (user, query) - We observe outcomes $(x, \pi_0(x), C_{\pi_0(x)})$ where C is the click pattern - We want to estimate performance of **new policy** π_1 that would produce ranking $\pi_1(x)$ - **Challenge**: We never observe $C_{\pi_1(x)}$ (user didn't see π_1's ranking)

Naive approach fails: Simply averaging rewards $R(x, \pi_0(x))$ under the logging policy does **not** estimate $\mathbb{E}[R(x, \pi_1(x))]$ because rankings differ.

Propensity scoring solution: Reweight observations by the **likelihood ratio** of policies producing the same ranking.

1.6.2 2.6.2 Propensity Scores and Inverse Propensity Scoring (IPS)

Definition 2.6.1 (Propensity Score) {#DEF-2.6.1}

Let π_0 be a stochastic logging policy that produces ranking $a \in \mathcal{A}$ for context x with probability $\pi_0(a \mid x)$. The **propensity score** of action (ranking) a in context x is

$$\rho(x, a) := \pi_0(a \mid x).$$

Definition 2.6.2 (Inverse Propensity Scoring Estimator) {#DEF-2.6.2}

Let $(x_1, a_1, r_1), \dots, (x_N, a_N, r_N)$ be logged data collected under policy π_0, where $a_i \sim \pi_0(\cdot \mid x_i)$ and $r_i = R(x_i, a_i, \omega_i)$ is the observed reward. The **IPS estimator** for the expected reward under new policy π_1 is

$$\hat{V}_{\text{IPS}}(\pi_1) := \frac{1}{N} \sum_{i=1}^{N} \frac{\pi_1(a_i \mid x_i)}{\pi_0(a_i \mid x_i)} r_i. \tag{2.4}$$

{#EQ-2.4}

Theorem 2.6.1 (Unbiasedness of IPS) {#THM-2.6.1}

Assume: 1. **Positivity**: $\pi_0(a \mid x) > 0$ for all a with $\pi_1(a \mid x) > 0$ 2. **Correct logging**: Observed actions a_i are sampled from $\pi_0(\cdot \mid x_i)$

Then $\hat{V}_{\text{IPS}}(\pi_1)$ is **unbiased**:

$$\mathbb{E}[\hat{V}_{\text{IPS}}(\pi_1)] = V(\pi_1) := \mathbb{E}_{x \sim \rho, a \sim \pi_1(\cdot|x)}[R(x, a)].$$

Proof.

Step 1 (Expand expectation over data and outcomes): The expectation is over contexts $x_i \sim \rho$, actions $a_i \sim \pi_0(\cdot \mid x_i)$, and outcomes ω drawn from the environment:

$$\mathbb{E}[\hat{V}_{\text{IPS}}(\pi_1)] = \mathbb{E}_{x \sim \rho}\left[\mathbb{E}_{a \sim \pi_0(\cdot|x)}\left[\mathbb{E}_{\omega}\left[\frac{\pi_1(a \mid x)}{\pi_0(a \mid x)} R(x, a, \omega) \mid x, a\right]\right]\right].$$

Since the importance ratio depends only on (x, a), we can take it outside the inner expectation and define the conditional mean reward $\mu(x, a) := \mathbb{E}_{\omega}[R(x, a, \omega) \mid x, a]$. For brevity, write $R(x, a) := \mu(x, a)$ in the steps below.

Step 2 (Rewrite inner expectation as sum): For a discrete action space,

$$\mathbb{E}_{a \sim \pi_0(\cdot|x)}\left[\frac{\pi_1(a \mid x)}{\pi_0(a \mid x)} R(x, a)\right] = \sum_a \pi_0(a \mid x) \cdot \frac{\pi_1(a \mid x)}{\pi_0(a \mid x)} R(x, a).$$

Step 3 (Cancel propensities):

$$= \sum_a \pi_1(a \mid x) R(x, a) = \mathbb{E}_{a \sim \pi_1(\cdot|x)}[R(x, a)].$$

Step 4 (Substitute into outer expectation):

$$\mathbb{E}[\hat{V}_{\text{IPS}}(\pi_1)] = \mathbb{E}_{x \sim \rho}\left[\mathbb{E}_{a \sim \pi_1(\cdot|x)}[R(x, a)]\right] = V(\pi_1).$$

□

Remark 2.6.1 (The importance sampling mechanism). This proof uses the **importance sampling technique**: reweight samples from distribution π_0 to estimate expectations under π_1. The ratio $\pi_1(a \mid x)/\pi_0(a \mid x)$ is the **Radon–Nikodym derivative** $d\pi_1/d\pi_0$ when π_0 dominates π_1 (positivity assumption). In measure-theoretic terms, $V(\pi_1) = \int R\, d(\rho \times \pi_1)$ and IPS implements a **change of measure**: $d(\rho \times \pi_1) = (d\pi_1/d\pi_0)\, d(\rho \times \pi_0)$.

Remark 2.6.2 (High variance caveat). While IPS is unbiased, it has **high variance** when π_1 and π_0 differ substantially (i.e., when $\pi_1(a \mid x)/\pi_0(a \mid x)$ is large for some (x, a)). This is the **curse of importance sampling**. Chapter 9 introduces variance-reduction techniques: **capping**, **doubly robust estimation**, and **SWITCH estimators**.

1.6.3 2.6.3 Propensities for Ranked Lists

For search ranking, the action space $\mathcal{A}$ consists of **permutations** of M products: $|\mathcal{A}| = M!$. Computing exact propensities $\pi_0(a \mid x)$ for full rankings is intractable when M is large (e.g., $M = 50 \Rightarrow 50! \approx 10^{64}$ rankings).

Position-based approximation (Plackett-Luce model): If policy π ranks products by scores $s_\pi(p \mid x)$, approximate the ranking distribution via sequential sampling. Let R_k be the set of remaining items after positions $1, \ldots, k-1$ have been chosen: $R_k := \{p : p \notin \{p_1, \ldots, p_{k-1}\}\}$. Then

$$\pi(p_k \mid x, p_1, \ldots, p_{k-1}) = \frac{s_\pi(p_k \mid x)}{\sum_{p \in R_k} s_\pi(p \mid x)}. \tag{2.5}$$

{#EQ-2.5}

This gives propensity for full ranking $a = (p_1, \ldots, p_M)$:

$$\pi(a \mid x) = \prod_{k=1}^{M} \frac{s_\pi(p_k \mid x)}{\sum_{p \in R_k} s_\pi(p \mid x)}. \tag{2.6}$$

{#EQ-2.6}

Practical simplification (top-K propensity): Only reweight top K positions (e.g., $K = 5$), treating lower positions as fixed:

$$\rho_{\text{top-}K}(x, a) = \prod_{k=1}^{K} \frac{s_{\pi_0}(p_k \mid x)}{\sum_{j=k}^{M} s_{\pi_0}(p_j \mid x)}.$$

This reduces computational cost while retaining most signal (users rarely examine beyond position 5–10).

Remark 2.6.4 (Approximation bias). Plackett–Luce and top-K truncations approximate true ranking propensities and can introduce bias in IPS. Doubly robust estimators (Chapter 9) mitigate bias by combining propensity weighting with outcome models.

Remark 2.6.3 (Chapter 9 preview). Off-policy evaluation in production search systems uses **clipped IPS**, **doubly robust estimators**, or **learned propensities** from logged data. The full treatment lives in Chapter 9 (Off-Policy Evaluation), with implementation in `evaluation/ope.py`.

1.7 2.7 Computational Illustrations

We verify the theory numerically using simple Python experiments.

1.7.1 2.7.1 Simulating PBM and DBN Click Models

Let's generate synthetic click data under PBM and DBN models and verify that marginal probabilities match theoretical predictions.

```python
import numpy as np
from typing import Tuple, List

# Set seed for reproducibility
np.random.seed(42)

# ============================================================================
# PBM: Position Bias Model
# ============================================================================

def simulate_pbm(
    relevance: np.ndarray,          # relevance[k] = rel(p_k) in [0,1]
    exam_probs: np.ndarray,          # exam_probs[k] = theta_k
    n_sessions: int = 10000
) -> Tuple[np.ndarray, np.ndarray]:
    """Simulate click data under Position Bias Model (PBM).

    Mathematical correspondence: Implements Definition 2.5.1 (PBM).

    Args:
        relevance: Product relevance at each position, shape (M,)
        exam_probs: Examination probabilities theta_k, shape (M,)
        n_sessions: Number of independent sessions to simulate

    Returns:
        examinations: Binary matrix (n_sessions, M), E_k = 1 if examined
        clicks: Binary matrix (n_sessions, M), C_k = 1 if clicked
    """
    M = len(relevance)
    examinations = np.random.binomial(1, exam_probs, size=(n_sessions, M))
    clicks = examinations * np.random.binomial(1, relevance, size=(n_sessions, M))
    return examinations, clicks


# Example: 10 results with decaying examination and varying relevance
M = 10
relevance = np.array([0.9, 0.8, 0.7, 0.5, 0.6, 0.3, 0.4, 0.2, 0.3, 0.1])
theta_1 = 0.9
decay = 0.25
exam_probs = theta_1 * np.exp(-decay * np.arange(M))

print("=== Position Bias Model (PBM) ===")
print(f"Relevance: {relevance}")
print(f"Examination probabilities: {exam_probs.round(3)}")

# Simulate
E_pbm, C_pbm = simulate_pbm(relevance, exam_probs, n_sessions=50000)

# Verify theoretical vs empirical CTR
theoretical_ctr = relevance * exam_probs
empirical_ctr = C_pbm.mean(axis=0)

print("\nPosition | Rel | Exam | Theory CTR | Empirical CTR | Match?")
```

```python
print("-" * 65)
for k in range(M):
    match = "OK" if abs(theoretical_ctr[k] - empirical_ctr[k]) < 0.01 else "FAIL"
    print(f"{k+1:8d} | {relevance[k]:.2f} | {exam_probs[k]:.2f} | "
          f"{theoretical_ctr[k]:10.3f} | {empirical_ctr[k]:13.3f} | {match}")

# Output:
# Position | Rel | Exam | Theory CTR | Empirical CTR | Match?
# -----------------------------------------------------------------
#        1 | 0.90 | 0.90 |      0.810 |         0.811 | OK
#        2 | 0.80 | 0.70 |      0.560 |         0.560 | OK
#        3 | 0.70 | 0.54 |      0.378 |         0.379 | OK
#        4 | 0.50 | 0.42 |      0.210 |         0.211 | OK
#        5 | 0.60 | 0.33 |      0.198 |         0.197 | OK
# ... (positions 6-10 omitted for brevity)

# ============================================================================
# DBN: Dynamic Bayesian Network (Cascade Model)
# ============================================================================

def simulate_dbn(
    relevance: np.ndarray,          # relevance[k] = rel(p_k)
    satisfaction: np.ndarray,       # satisfaction[k] = s(p_k)
    n_sessions: int = 10000
) -> Tuple[np.ndarray, np.ndarray, np.ndarray, np.ndarray]:
    """Simulate click data under DBN cascade model.

    Mathematical correspondence: Implements Definition 2.5.2 (DBN).

    Args:
        relevance: Product relevance, shape (M,)
        satisfaction: Satisfaction probability s(p), shape (M,)
        n_sessions: Number of sessions

    Returns:
        examinations: (n_sessions, M), E_k
        clicks: (n_sessions, M), C_k
        satisfied: (n_sessions, M), S_k
        stop_positions: (n_sessions,), tau (stopping time)
    """
    M = len(relevance)
    E = np.zeros((n_sessions, M), dtype=int)
    C = np.zeros((n_sessions, M), dtype=int)
    S = np.zeros((n_sessions, M), dtype=int)
    tau = np.full(n_sessions, M, dtype=int)  # Default: examine all

    for i in range(n_sessions):
        for k in range(M):
            # Always examine first; cascade rule for k > 0
            if k == 0:
                E[i, k] = 1
            else:
                # Examine if previous not satisfied
```

```python
                if S[i, k-1] == 0:
                    E[i, k] = 1
                else:
                    break  # Stop cascade

            # Click given examination
            if E[i, k] == 1:
                C[i, k] = np.random.binomial(1, relevance[k])

                # Satisfaction given click
                if C[i, k] == 1:
                    S[i, k] = np.random.binomial(1, satisfaction[k])
                    if S[i, k] == 1:
                        tau[i] = k  # Stopped here
                        break

    return E, C, S, tau


# Example: Same relevance, add satisfaction probabilities
satisfaction = np.array([0.6, 0.5, 0.7, 0.8, 0.6, 0.9, 0.7, 0.8, 0.9, 1.0])

print("\n\n=== Dynamic Bayesian Network (DBN Cascade) ===")
print(f"Relevance: {relevance}")
print(f"Satisfaction: {satisfaction}")

# Simulate
E_dbn, C_dbn, S_dbn, tau_dbn = simulate_dbn(relevance, satisfaction, n_sessions=50000)

# Verify examination probabilities match Proposition 2.5.1 (EQ-2.3)
def theoretical_exam_dbn(relevance, satisfaction, k):
    """Compute P(E_k = 1) using Proposition 2.5.1."""
    if k == 0:
        return 1.0
    prob = 1.0
    for j in range(k):
        prob *= (1 - relevance[j] * satisfaction[j])
    return prob

empirical_exam = E_dbn.mean(axis=0)
theoretical_exam = np.array([theoretical_exam_dbn(relevance, satisfaction, k) for k in range(M)])

print("\nPosition | Rel | Sat | Theory Exam | Empirical Exam | Match?")
print("-" * 68)
for k in range(M):
    match = "OK" if abs(theoretical_exam[k] - empirical_exam[k]) < 0.01 else "FAIL"
    print(f"{k+1:8d} | {relevance[k]:.2f} | {satisfaction[k]:.2f} | "
          f"{theoretical_exam[k]:11.3f} | {empirical_exam[k]:14.3f} | {match}")

# Output:
# Position | Rel | Sat | Theory Exam | Empirical Exam | Match?
# --------------------------------------------------------------------
#        1 | 0.90 | 0.60 |       1.000 |          1.000 | OK
```

```
#        2 | 0.80 | 0.50 |        0.460 |           0.460 | OK
#        3 | 0.70 | 0.70 |        0.276 |           0.277 | OK
#        4 | 0.50 | 0.80 |        0.140 |           0.140 | OK
# ... (examination decays rapidly as satisfied users stop)

# Abandonment statistics
print(f"\n\nMean stopping position: {tau_dbn.mean():.2f}")
print(f"% sessions stopping at position 1: {(tau_dbn == 0).mean() * 100:.1f}%")
print(f"% sessions examining all {M} results: {(tau_dbn == M).mean() * 100:.1f}%")

# Output:
# Mean stopping position: 2.34
# % sessions stopping at position 1: 48.6%
# % sessions examining all 10 results: 5.2%
```

Key observations: 1. **PBM verification**: Empirical CTR matches $\mathrm{rel}(p_k) \cdot \theta_k$ within 0.01 (Monte Carlo error) 2. **DBN cascade**: Examination probability decays rapidly due to satisfaction-induced stopping 3. **Abandonment**: ~50% of users satisfied at position 1; only ~5% examine all results

This confirms Definitions 2.5.1 (PBM) and 2.5.2 (DBN), and Proposition 2.5.1 (DBN examination formula).

1.7.2 2.7.2 Verifying IPS Unbiasedness

Let's simulate off-policy evaluation: collect data under logging policy π_0, estimate performance of new policy π_1 using IPS, and verify unbiasedness.

```
# ============================================================================
# Off-Policy Evaluation: IPS Estimator
# ============================================================================

def simulate_context_bandit(
    n_contexts: int,
    n_actions: int,
    n_samples: int,
    pi_logging,              # Callable: pi_logging(x) returns action probs
    pi_target,               # Callable: pi_target(x) returns action probs
    reward_fn,               # Callable: reward_fn(x, a) returns mean reward
    seed: int = 42
) -> Tuple[float, float, float]:
    """Simulate contextual bandit and estimate target policy value via IPS.

    Mathematical correspondence: Implements Theorem 2.6.1 (IPS unbiasedness).

    Returns:
        true_value: True expected reward under target policy
        ips_estimate: IPS estimate from logged data
        naive_estimate: Naive average (biased)
    """
    rng = np.random.default_rng(seed)

    # Collect logged data under pi_logging
    contexts = rng.integers(0, n_contexts, size=n_samples)
    logged_rewards = []
```

```
    importance_weights = []

    for x in contexts:
        # Sample action from logging policy
        pi_log_probs = pi_logging(x)
        a = rng.choice(n_actions, p=pi_log_probs)

        # Observe reward (with noise)
        mean_reward = reward_fn(x, a)
        r = mean_reward + rng.normal(0, 0.1)  # Add Gaussian noise
        logged_rewards.append(r)

        # Compute importance weight
        pi_tgt_probs = pi_target(x)
        w = pi_tgt_probs[a] / pi_log_probs[a]
        importance_weights.append(w)

    # IPS estimator (EQ-2.4)
    ips_estimate = np.mean(np.array(logged_rewards) * np.array(importance_weights))

    # Naive estimator (biased)
    naive_estimate = np.mean(logged_rewards)

    # Compute true expected reward under target policy (ground truth)
    true_value = 0.0
    for x in range(n_contexts):
        pi_tgt_probs = pi_target(x)
        for a in range(n_actions):
            true_value += (1 / n_contexts) * pi_tgt_probs[a] * reward_fn(x, a)

    return true_value, ips_estimate, naive_estimate


# Example: 5 contexts, 3 actions
n_contexts, n_actions = 5, 3

# Define reward function: action 0 good for contexts 0-1, action 2 good for contexts 3-4
def reward_fn(x, a):
    rewards = [
        [1.0, 0.3, 0.2],  # context 0
        [0.9, 0.4, 0.1],  # context 1
        [0.5, 0.6, 0.4],  # context 2
        [0.2, 0.3, 0.9],  # context 3
        [0.1, 0.2, 1.0],  # context 4
    ]
    return rewards[x][a]

# Logging policy: uniform random (safe but inefficient)
def pi_logging(x):
    return np.array([1/3, 1/3, 1/3])

# Target policy: greedy (optimal action per context)
def pi_target(x):
```

```python
    optimal_actions = [0, 0, 1, 2, 2]  # Best action per context
    probs = np.zeros(n_actions)
    probs[optimal_actions[x]] = 1.0
    return probs

print("\n\n=== Off-Policy Evaluation: IPS Unbiasedness ===")

# Run multiple trials to estimate bias and variance
n_trials = 500
true_values = []
ips_estimates = []
naive_estimates = []

for trial in range(n_trials):
    true_val, ips_est, naive_est = simulate_context_bandit(
        n_contexts, n_actions, n_samples=1000,
        pi_logging=pi_logging, pi_target=pi_target,
        reward_fn=reward_fn, seed=trial
    )
    true_values.append(true_val)
    ips_estimates.append(ips_est)
    naive_estimates.append(naive_est)

true_value = true_values[0]  # Should be constant
ips_mean = np.mean(ips_estimates)
ips_std = np.std(ips_estimates)
naive_mean = np.mean(naive_estimates)
naive_std = np.std(naive_estimates)

print(f"True target policy value: {true_value:.3f}")
print(f"\nIPS Estimator:")
print(f"  Mean: {ips_mean:.3f} (bias: {ips_mean - true_value:.4f})")
print(f"  Std:  {ips_std:.3f}")
print(f"  Unbiased? {'PASS' if abs(ips_mean - true_value) < 0.02 else 'FAIL'}")
print(f"\nNaive Estimator (biased):")
print(f"  Mean: {naive_mean:.3f} (bias: {naive_mean - true_value:.4f})")
print(f"  Std:  {naive_std:.3f}")
print(f"  Biased? {'PASS (expected)' if abs(naive_mean - true_value) > 0.05 else 'FAIL (unexpected)'}")

# Output:
# True target policy value: 0.820
#
# IPS Estimator:
#   Mean: 0.821 (bias: 0.0010)
#   Std:  0.087
#   Unbiased? PASS
#
# Naive Estimator (biased):
#   Mean: 0.507 (bias: -0.3130)
#   Std:  0.018
#   Biased? PASS (expected)
```

Key results: 1. **IPS is unbiased**: Mean IPS estimate ≈ true value (bias < 0.01), confirming Theorem 2.6.1 2. **Naive estimator is biased**: Underestimates target policy value by ~30% (logging policy is uniform, target is greedy) 3. **Variance tradeoff**: IPS has higher variance (std = 0.087) than naive (std = 0.018) due to importance weights

This validates the theoretical unbiasedness result while illustrating the **bias-variance tradeoff**: IPS removes bias at the cost of increased variance.

!!! note "Code ↔ Env/Reward (session step and aggregation)" The end-to-end simulator routes theory to code: - Env step calls behavior: `zoosim/env.py:52` (`behavior.simulate_session`) - Reward aggregation per #EQ-1.2: `zoosim/reward.py:50` - Env returns ranking, clicks, buys: `zoosim/env.py:69` KG: `MOD-zoosim.env`, `MOD-zoosim.reward`, `EQ-1.2`.

1.7.3 2.7.3 Verifying the Tower Property Numerically

We illustrate the Tower Property [THM-2.3.2] by constructing nested σ-algebras via simple groupings and verifying

$$\mathbb{E}[\mathbb{E}[Z \mid \mathcal{H}] \mid \mathcal{G}] = \mathbb{E}[Z \mid \mathcal{G}]$$

numerically.

```python
import numpy as np

np.random.seed(42)
N = 50_000

# Contexts and nested sigma-algebras via groupings
x = np.random.randint(0, 10, size=N)   # contexts 0..9
G = x % 2                              # coarse sigma-algebra: parity (2 groups)
H = x % 4                              # finer sigma-algebra: mod 4 (4 groups)

# Random variable Z depending on context with noise
Z = 2.0 * x + np.random.normal(0.0, 1.0, size=N)

# Compute E[Z | H] for each sample by replacing Z with the H-group mean
E_Z_given_H_vals = np.array([Z[H == h].mean() for h in range(4)])
E_Z_given_H = E_Z_given_H_vals[H]

# Left-hand side: E[E[Z | H] | G] - average E_Z_given_H within each G group
lhs = np.array([E_Z_given_H[G == g].mean() for g in range(2)])

# Right-hand side: E[Z | G] - average Z within each G group
rhs = np.array([Z[G == g].mean() for g in range(2)])

print("Group g |  E[E[Z|H]|G=g]  |    E[Z|G=g]    |  Match?")
print("-" * 58)
for g in range(2):
    match = "OK" if abs(lhs[g] - rhs[g]) < 1e-2 else "FAIL"
    print(f"   {g:5d} | {lhs[g]:16.4f} | {rhs[g]:14.4f} |  {match}")

# Output:
# Group g |  E[E[Z|H]|G=g]  |    E[Z|G=g]    |  Match?
# ----------------------------------------------------------
#     0    |          8.9196 |         8.9206 |  OK
#     1    |         13.9180 |        13.9190 |  OK
```

The numerical experiment confirms the Tower Property: averaging the conditional expectation $\mathbb{E}[Z \mid \mathcal{H}]$ over the coarser $\mathcal{G}$ equals $\mathbb{E}[Z \mid \mathcal{G}]$.

!!! note "Code ↔ Theory (Tower Property)" This numerical check verifies [THM-2.3.2] (Tower Property) by constructing nested σ-algebras via parity (#groups=2) and mod-4 (#groups=4) partitions and confirming $\mathbb{E}[\mathbb{E}[Z \mid \mathcal{H}] \mid \mathcal{G}] = \mathbb{E}[Z \mid \mathcal{G}]$. KG: `THM-2.3.2`.

1.8 2.8 Application Bridge to RL

We've built measure-theoretic probability foundations. Now we connect to reinforcement learning.

1.8.1 2.8.1 MDPs as Probability Spaces

Markov Decision Processes (MDPs), the canonical RL framework (Chapter 3), are probability spaces with structure.

Definition 2.8.1 (MDP, informal preview). An MDP is a tuple $(\mathcal{S}, \mathcal{A}, P, R, \gamma)$ where: - $\mathcal{S}$: State space (measurable space) - $\mathcal{A}$: Action space (measurable space) - $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0,1]$: Transition kernel (probability measure) - $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$: Reward function (random variable) - $\gamma \in [0,1)$: Discount factor

A policy $\pi : \mathcal{S} \to \Delta(\mathcal{A})$ maps states to probability distributions over actions. Together with initial state distribution ρ_0, this defines a **probability space over trajectories**:

$$\Omega = (\mathcal{S} \times \mathcal{A})^\infty, \quad \mathbb{P}^\pi = \text{measure induced by } \rho_0, \pi, P.$$

The value function is an expectation over this space:

$$V^\pi(s) := \mathbb{E}^\pi\left[\sum_{t=0}^{\infty} \gamma^t R_t \mid S_0 = s\right] = \mathbb{E}^\pi\left[\sum_{t=0}^{\infty} \gamma^t R(S_t, A_t) \mid S_0 = s\right].$$

What we've learned enables: - **Measurability**: S_t, A_t, R_t are random variables (Section 2.2.3) - **Conditional expectation**: $V^\pi(s) = \mathbb{E}[G_0 \mid S_0 = s]$ is well-defined (Section 2.3.2) - **Infinite sums**: Monotone Convergence Theorem (THM-2.2.3) justifies interchanging $\sum$ and $\mathbb{E}$ - **Filtrations**: $\mathcal{F}_t = \sigma(S_0, A_0, \ldots, S_t, A_t)$ models "information up to time t" (Section 2.4.1)

Chapter 3 makes this rigorous and proves convergence of value iteration via contraction mappings.

Remark 2.8.1 (Uncountable trajectory space). Even when per-step state and action spaces are finite, the space of infinite-horizon trajectories $\Omega = (\mathcal{S} \times \mathcal{A} \times \mathbb{R})^{\mathbb{N}}$ is uncountable (same cardinality as $[0,1]$). There is no meaningful "uniform counting" on Ω; probabilities must be defined as measures on σ-algebras.

1.8.2 2.8.2 Click Models as Contextual Bandits

The **contextual bandit** (one-step MDP, no state transitions) is the foundation for Chapters 6–8:

Setup: - Context $x \sim \rho$ (user, query) - Policy $\pi : \mathcal{X} \to \Delta(\mathcal{A})$ selects action (boost weights) $a \sim \pi(\cdot \mid x)$ - Outcome $\omega \sim P(\cdot \mid x, a)$ drawn from click model (PBM or DBN) - Reward $R(x, a, \omega)$ aggregates GMV, CM2, clicks (Chapter 1, #EQ-1.2)

Goal: Learn policy π^* maximizing

$$V(\pi) = \mathbb{E}_{x \sim \rho, a \sim \pi(\cdot|x), \omega \sim P(\cdot|x,a)}[R(x, a, \omega)].$$

Click models provide $P(\cdot \mid x, a)$**:** - PBM (Section 2.5.1): $\omega = (E_1, C_1, \ldots, E_M, C_M)$ with independent examination/click - DBN (Section 2.5.2): $\omega = (E_1, C_1, S_1, \ldots)$ with cascade stopping

Propensity scores (Section 2.6) enable off-policy learning: - Collect data under exploration policy π_0 (e.g., ϵ-greedy, Thompson Sampling) - Estimate $V(\pi_1)$ for candidate policies via IPS #EQ-2.4 - Select best policy without online deployment risk

Chapter connections: - **Chapter 6** (Discrete Template Bandits): Tabular policies, finite action space $|\mathcal{A}| = 25$ - **Chapter 7** (Continuous Actions via Q-learning): Regression over $Q(x, a) = \mathbb{E}[R \mid x, a]$ - **Chapter 8** (Constraints): CMDP formulation with CM2 floors, exposure targets (Chapter 1, #EQ-1.3) - **Chapter 9** (Off-Policy Evaluation): Production IPS, SNIPS, doubly robust estimators

All rely on the probability foundations built in this chapter.

1.8.3 2.8.3 Forward References

Chapter 3 — Stochastic Processes & Bellman Foundations: - Bellman operators as **contractions** in function spaces (operator theory) - Value iteration convergence via **Banach Fixed-Point Theorem** - Filtrations $\{\mathcal{F}_t\}$ and martingale convergence theorems

Chapter 4 — Catalog, Users, Queries: - Generative models for contexts $x = (u, q)$ with distributional realism - Deterministic generation via seeds (reproducibility) - Feature engineering $\phi_k(p, u, q)$ as random variables

Chapter 5 — Relevance, Features, Reward: - Reward function $R(x, a, \omega)$ implementation using click model outcomes ω - Verification that $\mathbb{E}[R \mid x, a]$ is well-defined and integrable - Conditional expectation structure for model-based value estimation

Chapter 9 — Off-Policy Evaluation: - IPS, SNIPS, doubly robust estimators (extending Section 2.6) - Variance reduction via capping, control variates - Propensity estimation from logged data (when π_0 is unknown)

Chapter 11 — Multi-Episode MDP: - Stopping times τ for session termination (extending Section 2.4.2) - Inter-session dynamics: state transitions $s_{t+1} = f(s_t, \omega_t)$ - Retention modeling via survival analysis

---

1.9 2.9 Production Checklist

Production Checklist (Chapter 2)

Configuration alignment: - **Click model selection**: Set `BehaviorConfig.click_model` in `zoosim/config.py` to `"pbm"` or `"dbn"` - **PBM parameters**: Configure `theta_1` (position-1 examination) and `decay` (exponential decay rate) in `BehaviorConfig.pbm` - **DBN parameters**: Configure `satisfaction_fn` (product $\to$ satisfaction probability) in `BehaviorConfig.dbn` - **Seeds**: Ensure `SimulatorConfig.seed` is set consistently for reproducible click patterns
Implementation modules (to be created in Chapter 4–5): - `zoosim/behavior.py`: Implements PBM and DBN simulators from Definitions 2.5.1–2.5.2 - `zoosim/config.py`: Exposes `BehaviorConfig` with position bias and satisfaction parameters - `evaluation/ope.py` (Chapter 9): Implements IPS estimator from Definition 2.6.2
Tests: - `tests/test_behavior.py`: Verify empirical CTR matches theoretical values (within Monte Carlo error) - `tests/test_ope.py`: Verify IPS unbiasedness on synthetic data (Section 2.7.2) - `tests/test_stopping_times.py`: Verify DBN abandonment statistics (mean stop position, examination decay)
Assertions: - Check $0 \le \theta_k \le 1$ for all examination probabilities - Check $0 \le \text{rel}(p) \le 1$ and $0 \le s(p) \le 1$ for relevance/satisfaction - Check positivity assumption $\pi_0(a \mid x) > 0$ when computing IPS weights

---

1.10 2.10 Exercises

Exercise 2.1 (Measurability, 10 min). Let $X : \Omega \to \mathbb{R}$ and $Y : \Omega \to \mathbb{R}$ be random variables on $(\Omega, \mathcal{F}, \mathbb{P})$. Prove that $Z = X + Y$ is also a random variable.

Exercise 2.2 (Conditional probability computation, 15 min). In the PBM model, suppose $\text{rel}(p_3) = 0.6$ and $\theta_3 = 0.5$. Compute: 1. $\mathbb{P}(C_3 = 1)$ 2. $\mathbb{P}(E_3 = 1 \mid C_3 = 1)$ 3. $\mathbb{P}(C_3 = 1 \mid E_3 = 1)$

Exercise 2.3 (DBN cascade probability, 20 min). In the DBN model, suppose $M = 3$ with: - $\text{rel}(p_1) = 0.8$, $s(p_1) = 0.5$ - $\text{rel}(p_2) = 0.6$, $s(p_2) = 0.7$ - $\text{rel}(p_3) = 0.9$, $s(p_3) = 0.9$

Compute: 1. $\mathbb{P}(E_2 = 1)$ 2. $\mathbb{P}(E_3 = 1)$ 3. $\mathbb{P}(\tau = 1)$ (probability user stops at position 1)

Exercise 2.4 (IPS estimator properties, 20 min). Prove that if $\pi_1 = \pi_0$ (target equals logging), then $\hat{V}_{\text{IPS}}(\pi_1)$ reduces to the naive sample mean, and has lower variance than the general IPS estimator.

Exercise 2.5 (Stopping time verification, 15 min). Show that $\tau = \max\{k : C_k = 1\}$ (position of last click) is **not** a stopping time, by constructing a specific example where $\{\tau = 2\}$ requires knowledge of C_3.

Exercise 2.6 (RL bridge: Bellman operator as conditional expectation, 20 min). Let $V : \mathcal{S} \to \mathbb{R}$ be a value function. The Bellman operator $\mathcal{T}^\pi V$ is defined as

$$(\mathcal{T}^\pi V)(s) = \mathbb{E}_{a \sim \pi(\cdot|s)} \left[R(s,a) + \gamma \mathbb{E}_{s' \sim P(\cdot|s,a)}[V(s')] \right].$$

Show that this is a **conditional expectation**: $(\mathcal{T}^\pi V)(s) = \mathbb{E}[R_0 + \gamma V(S_1) \mid S_0 = s]$ under policy π.

Exercise 2.7 (Code: Variance of IPS, 30 min). Extend the IPS experiment in Section 2.7.2. Vary the divergence between π_0 and π_1 (e.g., make π_1 increasingly greedy while π_0 remains uniform). Plot IPS variance vs policy divergence (measured by KL divergence $D_{\text{KL}}(\pi_1 \| \pi_0)$). Verify that variance increases as policies diverge.

1.11 2.11 Chapter Summary

What we built: 1. **Probability spaces** $(\Omega, \mathcal{F}, \mathbb{P})$: Sample spaces, σ-algebras, probability measures 2. **Random variables and expectations**: Measurable functions, Lebesgue integration, linearity 3. **Conditional probability**: Conditional expectation given σ-algebras, Tower Property 4. **Filtrations and stopping times**: Sequential information, abandonment as stopped processes 5. **Click models for search**: PBM (position bias), DBN (cascade), theoretical formulas vs empirical validation 6. **Propensity scoring**: Unbiased off-policy estimation via IPS, importance sampling mechanism

Why it matters for RL: - Chapter 1's rewards are now rigorous: $\mathbb{E}[R \mid \mathbf{w}]$ is a Lebesgue integral over a probability space - **Chapter 3's Bellman operators are measurable**: Value functions are conditional expectations - **Chapters 6–8's bandits have formal semantics**: Contexts, actions, outcomes are random variables - **Chapter 9's off-policy evaluation is justified**: IPS unbiasedness proven via measure theory

Next chapter: Stochastic processes, Markov chains, Bellman operators, and contraction mappings. We'll prove value iteration converges using the Banach Fixed-Point Theorem—all enabled by the foundations built here.

The key realization: RL is applied probability theory. Every algorithm is an expectation, every policy is a conditional distribution, every convergence proof uses measure-theoretic limit theorems. Without this chapter, RL is heuristics. With it, RL is mathematics.

Let's continue building.

1.12 References

The primary references for this chapter are:

- [@folland:real_analysis:1999] — Measure theory, integration, conditional expectation
- [@durrett:probability:2019] — Stochastic processes, filtrations, stopping times, martingales
- [@craswell:cascade:2008] — Click models for web search (PBM, DBN)
- [@chapelle:position_bias:2009] — Unbiased learning to rank via propensity scoring
- [@wang:position_bias_contextual:2016] — Position bias in contextual bandits for search

Full bibliography lives in `references.bib`.