Chapter 2 — Probability, Measure, and Click Models

Vlad Prytula

2.1 Motivation: Why Search Needs Measure Theory

This chapter develops the measure-theoretic probability framework—σ-algebras, measurable spaces, and Radon–Nikodym derivatives—that underlies off-policy evaluation (OPE) and, more broadly, reinforcement learning on general state and action spaces.

A natural question is why such machinery is needed for ranking. The answer is that OPE is a change-of-measure argument. When we compute an importance weight

$$w_t = \frac{\pi(a \mid x)}{\mu(a \mid x)},$$

we are computing a Radon–Nikodym derivative. On continuous representations (e.g., embeddings), point probabilities are typically zero, so ratios must be defined at the level of measures, not by naive counting. The purpose of this chapter is to make these ratios and expectations mathematically well-defined so that later estimators are theorems rather than heuristics.

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. Chapter 1 used $\mathbb{E}[R \mid W]$ informally. Now we make it precise: expectations are **Lebesgue integrals** over probability measures, with $\mathbb{E}[R \mid W]$ a $\sigma(W)$-measurable random variable and regular versions $\mathbb{E}[R \mid W = w]$ defined when standard Borel assumptions hold. 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.

We begin.

How much measure theory is needed? (Reading guide)

Implementation-first reading: On a first pass, it is reasonable to skim §§2.2–2.4 (measure-theoretic foundations) and return when a later chapter invokes a specific result. The algorithm-facing core is:

- **§2.5 (Click Models)**: Position Bias Model and DBN—these directly parameterize user behavior in the simulator
- **§2.6 (Propensity Scoring)**: Foundation for importance weighting in off-policy evaluation (Chapter 9)
- **§2.7–2.8 (Computational Verification & RL Bridges)**: NumPy experiments and connections to MDP formalism

Theory-first reading: §§2.2–2.4 provide the rigorous foundations that

make policy gradient interchange (Chapter 8) and Radon–Nikodym importance weights (Chapter 9) theorems rather than heuristics. The proofs there are self-contained and follow Folland's *Real Analysis* [@folland:real_analysis:1999].
Navigation: If σ-algebras are heavy on a first reading, begin with §2.5 and return to §§2.2–2.4 as needed.

Assumptions (probability and RL foundations)

These assumptions apply throughout this chapter. - Spaces $\mathcal{S}$, $\mathcal{A}$, contexts $\mathcal{X}$, and outcome spaces are standard Borel (measurable subsets of Polish spaces, i.e. separable completely metrizable topological spaces). This guarantees existence of regular conditional probabilities and measurable stochastic kernels. - Rewards are integrable: $R \in L^1$. For discounted RL with $0 \leq \gamma < 1$, assume bounded rewards (or a uniform bound on expected discounted sums) so value iteration is well-defined. - Transition and policy kernels $P(\cdot \mid s, a)$ and $\pi(\cdot \mid s)$ are Markov kernels measurable in their arguments. - Off-policy evaluation (IPS): positivity/overlap - if $\pi(a \mid x) > 0$ then $\mu(a \mid x) > 0$ for the logging policy μ.

Background: Standard Borel and Polish Spaces

Polish space: A topological space that is separable (has a countable dense subset) and completely metrizable (admits a complete metric inducing its topology). Examples: $\mathbb{R}^n$, separable Hilbert spaces, the space of continuous functions $C([0,1])$, discrete countable sets.
Standard Borel space: A measurable space $(X, \mathcal{B})$ isomorphic (as a measurable space) to a Borel subset of a Polish space equipped with its Borel σ-algebra. Equivalently: a measurable space that "looks like" $\mathbb{R}$, $[0,1]$, $\mathbb{N}$, or a finite set from the measure-theoretic viewpoint.
Why this matters for RL: Standard Borel spaces enjoy three crucial properties:

1. **Regular conditional probabilities exist**: $\mathbb{P}(A \mid X = x)$ is well-defined as a function of x
2. **Measurable selection theorems apply**: Optimal policies $\pi^*(s) = \arg\max_a Q(s, a)$ are measurable functions
3. **Disintegration of measures**: Joint distributions factor cleanly into marginals and conditionals

Without these assumptions, pathological counterexamples exist where conditional expectations are undefined or optimal policies are non-measurable. The standard Borel assumption is the "fine print" that makes RL theory work.
Reference: [@kechris:classical_dsp:1995, Chapter 12] provides the definitive treatment. For RL applications, see [@bertsekas:stochastic_oc:1996,

Appendix C].

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.

2.2.1 Measurable Spaces and σ-Algebras

Definition 2.2.1 (Measurable Space)

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 $A_1, A_2, \ldots \in \mathcal{F}$, 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 $\Omega = \{\omega_1, \ldots, \omega_N\}$ 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).

Practical anchor: importance weights

In Chapter 9 we define the importance weight as a Radon-Nikodym derivative:

$$\rho = \frac{d\mathbb{P}^{\pi}}{d\mathbb{P}^{\mu}}.$$

Existence requires absolute continuity (overlap) $\mathbb{P}^{\pi} \ll \mathbb{P}^{\mu}$. See 2.3.5 for the canonical identity behind importance weighting.

2.2.2 Probability Measures

Definition 2.2.2 (Probability Measure)

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 $\Omega = \{1, 2, ..., N\}$, $\mathcal{F} = 2^{\Omega}$. 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 $(a, b) \subseteq [0, 1]$. **Carathéodory's Extension Theorem** [@folland:real_analysis:1999, Theorem 1.14] says that any countably additive set function defined on an algebra (here, finite unions of intervals) extends uniquely to the σ-algebra it generates. Applying it here extends $\mathbb{P}$ uniquely to all Borel sets, 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.

2.2.3 Random Variables

Definition 2.2.3 (Random Variable)

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 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) . 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, $X^{-1}(f^{-1}(A)) \in \mathcal{F}_1$. □

Remark 2.2.3 (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 2.2.4 (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.

2.2.4 Expectation and Integration

Definition 2.2.4 (Expectation)

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 $\Omega = \{\omega_1, \ldots, \omega_N\}$ 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)

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)

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 2.2.7 (Dominated convergence, informal). The **Dominated Convergence Theorem** complements monotone convergence: if $X_n \to X$ almost surely and there exists an integrable random variable Y with $|X_n| \leq Y$ for all n, then X is integrable and $\mathbb{E}[X_n] \to \mathbb{E}[X]$. Intuitively, a single integrable bound Y prevents "mass from escaping to infinity," allowing us to interchange limit and expectation. In later chapters this justifies moving gradients or limits inside expectations when rewards or score functions are uniformly bounded.

Remark 2.2.8 (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 2.2.3 allows us to interchange limits and expectations when computing Bellman operator fixed points (Chapter 3).

2.2.5 Measurable Functions

Definition 2.2.5 (Measurable Function)

Let $(E, \mathcal{E})$ and $(F, \mathcal{F})$ be measurable spaces. A function $f : E \to F$ is $(\mathcal{E}, \mathcal{F})$**-measurable** if for all $A \in \mathcal{F}$,

$$f^{-1}(A) := \{x \in E : f(x) \in A\} \in \mathcal{E}.$$

Remark 2.2.9 (Checking measurability via generators). If $\mathcal{F}$ is generated by a collection $\mathcal{G}$ (e.g., open intervals for Borel sets on $\mathbb{R}$), it suffices to check $f^{-1}(G) \in \mathcal{E}$ for all $G \in \mathcal{G}$.

Example 2.2.9 (Real-valued measurability). For $f : (E, \mathcal{E}) \to (\mathbb{R}, \mathcal{B}(\mathbb{R}))$, measurability is equivalent to $f^{-1}((-\infty, a)) \in \mathcal{E}$ for all $a \in \mathbb{R}$.

This definition justifies the **random variable** definition (2.2.3): a random variable is simply a measurable map from $(\Omega, \mathcal{F})$ into a codomain measurable space.

2.2.6 Segment Distributions (Finite Spaces)

Definition 2.2.6 (Segment distribution)

Let $\mathcal{S}_{\text{seg}} = \{s_1, \dots, s_K\}$ be a finite set of user segments equipped with the power-set σ-algebra $2^{\mathcal{S}_{\text{seg}}}$. A **segment distribution** is a probability measure $\mathbb{P}_{\text{seg}}$ on $\mathcal{S}_{\text{seg}}$. Equivalently, it is a probability vector $\mathbf{p}_{\text{seg}} \in \Delta_K$ such that $\mathbb{P}_{\text{seg}}(\{s_i\}) = (\mathbf{p}_{\text{seg}})_i$ and $\sum_{i=1}^{K} (\mathbf{p}_{\text{seg}})_i = 1$.

Remark 2.2.10 (Simulator connection). In our simulator, $\mathcal{S}_{\text{seg}}$ is the finite set of segment labels (e.g., `price_hunter`, `premium`), and $\mathbf{p}_{\text{seg}}$ is sampled by `zoosim/world/users.py::sample_user`.

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.

2.3.1 Conditional Probability Given an Event

Definition 2.3.1 (Conditional Probability)

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)

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).

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)

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)

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)

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]. Informally, Radon-Nikodym says that if a (finite) measure ν is absolutely continuous with respect to another measure $\mathbb{P}$, then there exists an integrable density h such that $\nu(A) = \int_A h \, d\mathbb{P}$ for all A; we write $h = d\nu/d\mathbb{P}$. 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 (Radon-Nikodym preview). Existence and uniqueness of conditional expectations ultimately rest on the Radon-Nikodym theorem 2.3.4. The same theorem yields the importance weights used in off-policy evaluation; see 2.3.5.

Code note: In `zoosim`, the IPS estimator computes `importance_weights` implementing the Radon-Nikodym weight from 2.3.5. If overlap fails - i.e., $\pi_0(a \mid x) = 0$ while $\pi_1(a \mid x) > 0$ - then the weight is undefined (formally infinite), and estimators based on it are ill-posed; implementations will either error or exhibit uncontrolled variance.

Theorem 2.3.4 (Radon-Nikodym)

Let μ and ν be σ-finite measures on $(\Omega, \mathcal{F})$ with $\nu \ll \mu$ (absolute continuity: $\mu(A) = 0 \Rightarrow \nu(A) = 0$ for all $A \in \mathcal{F}$). Then there exists a non-negative measurable function $f : \Omega \to [0, \infty)$ such that for all $A \in \mathcal{F}$:

$$\nu(A) = \int_A f \, d\mu.$$

The function f, unique μ-a.e., is called the **Radon-Nikodym derivative** and written $f = \frac{d\nu}{d\mu}$.

Proof. See [@folland:real_analysis:1999, Theorem 3.8]. □

Remark 2.3.5 (Importance sampling as change of measure) . Let μ and ν be probability measures on a measurable space $(\mathcal{A}, \mathcal{E})$ with $\nu \ll \mu$. For any μ-integrable function f,

$$\int_{\mathcal{A}} f(a)\, \nu(da) = \int_{\mathcal{A}} f(a)\, \frac{d\nu}{d\mu}(a)\, \mu(da).$$

In off-policy evaluation, for each fixed context x we take $\mu(\cdot) := \pi_0(\cdot \mid x)$ (logging policy) and $\nu(\cdot) := \pi_1(\cdot \mid x)$ (evaluation policy). The **importance weight** is the Radon-Nikodym derivative

$$\rho(a \mid x) := \frac{d\pi_1(\cdot \mid x)}{d\pi_0(\cdot \mid x)}(a),$$

which reduces to the ratio $\pi_1(a \mid x)/\pi_0(a \mid x)$ in the finite-action case. Absolute continuity $\pi_1(\cdot \mid x) \ll \pi_0(\cdot \mid x)$ is exactly the overlap condition: if $\pi_1(a \mid x) > 0$ then $\pi_0(a \mid x) > 0$.

Remark 2.3.6 (Conditioning on a random variable vs a value). We write $\mathbb{E}[R \mid W]$ for the $\sigma(W)$-measurable conditional expectation. When evaluating at a value w, we use a regular conditional distribution $\mathbb{P}(R \in \cdot \mid W = w)$ (which exists under the standard Borel assumption) and set

$$\mathbb{E}[R \mid W = w] := \int r\, d\mathbb{P}(R \in dr \mid W = w).$$

If w is a deterministic parameter (not a random variable), $\mathbb{E}[R \mid w]$ denotes a function of w rather than a conditional expectation in the measure-theoretic sense.

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**.

2.4.1 Filtrations

Definition 2.4.1 (Filtration)

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)

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").

2.4.2 Stopping Times

Definition 2.4.3 (Stopping Time)

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 $t \in \mathbb{N}$,

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

In discrete time this is equivalent to requiring $\{\tau = t\} \in \mathcal{F}_t$ for all 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)

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 the stopped σ-algebra

$$\mathcal{F}_\tau := \{A \in \mathcal{F} : A \cap \{\tau \leq t\} \in \mathcal{F}_t \text{ for all } t\}.$$

Proof.

Step 1 (Reduce to Borel preimages). 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 (Verify the defining condition). Fix $t \in \mathbb{N}$. We show $\{X_\tau \in B\} \cap \{\tau \leq t\} \in \mathcal{F}_t$. Decompose:

$$\{X_\tau \in B\} \cap \{\tau \leq t\} = \bigcup_{k=0}^{t} \left(\{\tau = k\} \cap \{X_k \in B\}\right).$$

Indeed, on $\{\tau = k\}$ we have $X_\tau = X_k$, and on $\{\tau \leq t\}$ only the indices $k \leq t$ contribute.

Step 3 (Measurability of each slice). Since τ is a stopping time, $\{\tau \leq k\} \in \mathcal{F}_k$ for each k. For $k = 0$ we have $\{\tau = 0\} = \{\tau \leq 0\} \in \mathcal{F}_0$. For $k \geq 1$,

$$\{\tau = k\} = \{\tau \leq k\} \setminus \{\tau \leq k-1\} \in \mathcal{F}_k.$$

By adaptation, $\{X_k \in B\} \in \mathcal{F}_k$. Therefore $\{\tau = k\} \cap \{X_k \in B\} \in \mathcal{F}_k \subseteq \mathcal{F}_t$ for each $k \leq t$.

Step 4 (Conclusion). The union in Step 2 is finite, hence lies in $\mathcal{F}_t$. Since this holds for all t, we have $\{X_\tau \in B\} \in \mathcal{F}_\tau$. □

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 2.2.3 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).

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.

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)

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) - $\mathrm{rel}(p_k)$: 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**:

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

   i.e.,

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

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

Click probability formula:

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

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 $\mathrm{rel}(p) = 0.5$ 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 (2.1); it becomes relevant for multi-position events.

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)

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**:

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

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

   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) . Under the DBN model, the probability of examining position k is

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

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 (2.2), $E_k = 1$ iff $E_{k-1} = 1$ and $S_{k-1} = 0$.

Step 3 (Probability of satisfaction given examination): Along the cascade, given examination at position j, satisfaction occurs iff the user clicks AND is satisfied given the click:

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

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

Step 4 (Chain rule via cascade): Since examination at position k requires not being satisfied at all $j < k$:

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

□

Remark 2.5.2 (Examination decay in DBN). By (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 stopping position

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

from Definition 2.5.2 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.

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). The production simulator `zoosim` implements a richer **Utility-Based Cascade Model** (§2.5.4), configurable via `zoosim/core/config.py`. This production model reproduces PBM's marginal factorization under a parameter specialization (Proposition 2.5.4) and exhibits cascade-style dependence driven by an internal state, analogous in spirit to DBN.

Remark 2.5.5 (Limitations relative to the simulator). PBM and DBN are analytically valuable but omit mechanisms that matter in `zoosim/dynamics/behavior.py`:

- **User heterogeneity**: segment-dependent preference parameters (price, private label, category affinities)
- **Continuous internal state**: a real-valued satisfaction/patience state rather than a binary stop indicator
- **Purchases and saturation**: purchase events and hard caps (e.g., max purchases) that influence termination
- **Query-typed position bias**: different position-bias curves by query class, not a single $\{\theta_k\}$

These omissions are benign for closed-form derivations (e.g., (2.1), (2.3)) but become first-order in production evaluation and simulation.

Code <-> Config (position bias and satisfaction)

PBM and DBN parameters map to configuration fields: - Examination bias vectors: `BehaviorConfig.pos_bias` in `zoosim/core/config.py:180-186` - Satisfaction dynamics: `BehaviorConfig.satisfaction_gain`, `BehaviorConfig.satisfaction_decay`, `BehaviorConfig.abandonment_threshold`, `BehaviorConfig.post_purchase_fatigue` in `zoosim/core/config.py:175-179`

Code <-> Behavior (Stopping Times & Satisfaction)

The abstract stopping time τ from 2.4.3 is implemented concretely in the user session loop:

- **Implementation**: `zoosim/dynamics/behavior.py` inside `simulate_session`
- **The Filtration**: The loop state (current `satisfaction`, `purchase_count`) represents $\mathcal{F}_t$.
- **The Stopping Rule**:
```
# The stopping time tau implementation
if rng.random() > examine_prob: break        # Abandonment (PBM/DBN)
if satisfaction < abandonment_threshold: break  # Satisfaction termination
if purchase_count >= purchase_limit: break      # Saturation
```
- **Integration**: The `SessionOutcome` returned is the stopped process X_τ.

2.5.4 The Utility-Based Cascade Model (Production Model)

The PBM and DBN models above are valuable for theoretical analysis and algorithm design, but our production simulator implements a richer model that combines the best of both paradigms with economically meaningful user preferences. This section formalizes the **Utility-Based Cascade Model** implemented in `zoosim/dynamics/behavior.py` and establishes its relationship to the textbook models.

Why formalize the production model? Without this bridge, students would learn theory (PBM/DBN) they cannot verify in the codebase, and practitioners would use code they cannot connect to guarantees. The core principle of this book—*theory and code in constant dialogue*—demands that our production simulator have rigorous mathematical foundations.

Definition 2.5.3 (Utility-Based Cascade Model)

Let user u have preference parameters $(\theta_{\text{price}}, \theta_{\text{pl}}, \theta_{\text{cat}})$ where $\theta_{\text{cat}} \in \mathbb{R}^{|\mathcal{C}|}$ encodes category affinities. For a ranking $(p_1, \dots, p_M)$ in response to query q, define the session dynamics:

1. **Latent utility at position k:**

$$U_k = \alpha_{\text{rel}} \cdot \text{match}(q, p_k) + \alpha_{\text{price}} \cdot \theta_{\text{price}} \cdot \log(1 + \text{price}_k) + \alpha_{\text{pl}} \cdot \theta_{\text{pl}} \cdot \mathbf{1}_{\text{is\_pl}(p_k)} + \alpha_{\text{cat}} \cdot \theta_{\text{cat}}(\text{cat}_k) + \varepsilon_k \tag{2.4}$$

   where $\text{match}(q, p_k) = \cos(\phi_q, \phi_{p_k})$ is semantic similarity (Chapter 5), cat_k is the category of product p_k, and $\varepsilon_k \overset{\text{iid}}{\sim} \mathcal{N}(0, \sigma_u^2)$ is utility noise.

2. **Click probability given examination:**

$$\mathbb{P}(C_k = 1 \mid E_k = 1, U_k) = \sigma(U_k) := \frac{1}{1 + e^{-U_k}} \tag{2.5}$$

3. **Examination probability (cascade with satisfaction):**

$$\mathbb{P}(E_k = 1 \mid \mathcal{F}_{k-1}) = \sigma(\text{pos\_bias}_k(q) + \beta_{\text{exam}} \cdot S_{k-1}) \tag{2.6}$$

   where $\text{pos\_bias}_k(q)$ depends on query type and position, and S_{k-1} is the running satisfaction state.

4. **Satisfaction dynamics:**

$$S_k = S_{k-1} + \gamma_{\text{gain}} \cdot U_k \cdot C_k - \gamma_{\text{decay}} \cdot (1 - C_k) - \gamma_{\text{fatigue}} \cdot B_k \tag{2.7}$$

   where $B_k \in \{0, 1\}$ indicates purchase at position k, and γ_{fatigue} captures post-purchase satiation.

5. **Stopping time:**

$$\tau = \min\{k : E_k = 0 \text{ or } S_k < \theta_{\text{abandon}} \text{ or } \sum_{j \le k} B_j \ge n_{\max}\} \tag{2.8}$$

   The session terminates at the first position where examination fails, satisfaction drops below threshold, or the purchase limit is reached.

Why this model? The Utility-Based Cascade captures three phenomena that pure PBM and DBN miss:

- **User heterogeneity**: Different users have different price sensitivities (θ_{price}), brand preferences (θ_{pl}), and category affinities (θ_{cat}). A premium user clicks expensive items; a price hunter clicks discounts. The utility structure (2.4) makes this heterogeneity explicit.
- **Satisfaction dynamics**: Unlike DBN's binary satisfaction, our running state S_k accumulates positive utility on clicks and decays on non-clicks. This models realistic shopping fatigue: a user who sees several irrelevant results becomes less likely to continue, even if they haven't yet clicked.
- **Purchase satiation**: The γ_{fatigue} term captures the observation that users who just bought something are less likely to continue browsing. This is economically significant for GMV optimization.

Proposition 2.5.4 (Nesting relations)

The Utility-Based Cascade Model contains PBM as a parameter specialization (at the level of per-position marginals) and, in general, induces cascade-style dependence through its internal state.

(a) PBM marginal factorization. Set $\alpha_{\text{price}} = \alpha_{\text{pl}} = \alpha_{\text{cat}} = 0$, $\sigma_u = 0$, $\beta_{\text{exam}} = 0$, $\gamma_{\text{gain}} = \gamma_{\text{decay}} = \gamma_{\text{fatigue}} = 0$, and $\theta_{\text{abandon}} = -\infty$ with $n_{\max} = \infty$ (no termination from purchases). Then for each position k,

$$\mathbb{P}(C_k = 1) = \theta_k \cdot \text{rel}(p_k),$$

where $\theta_k := \sigma(\text{pos\_bias}_k(q))$ and $\text{rel}(p_k) := \sigma(\alpha_{\text{rel}} \cdot \text{match}(q, p_k))$. This matches the PBM marginal formula (2.1) under the above identification.

(b) Cascade dependence (DBN-like mechanism). If $\beta_{\text{exam}} \neq 0$ and the state S_k evolves nontrivially, then the examination probability at position k depends on the past through S_{k-1}. In particular, in general the pairs (E_k, C_k) are not independent across positions. The DBN model is a discrete-state, hard-stopping cascade driven by a binary satisfaction variable; Definition 2.5.3 should be viewed as a smooth state-space cascade rather than a distributionally identical reduction.

Proof. For (a), under the stated specialization, Definition 2.5.3(3) yields $\mathbb{P}(E_k = 1 \mid \mathcal{F}_{k-1}) = \sigma(\text{pos\_bias}_k(q)) =: \theta_k$, independent of the past. Definition 2.5.3(1) makes $U_k = \alpha_{\text{rel}} \cdot \text{match}(q, p_k)$ deterministic, hence Definition 2.5.3(2) gives $\mathbb{P}(C_k = 1 \mid E_k = 1) = \sigma(U_k) =: \text{rel}(p_k)$. Therefore $\mathbb{P}(C_k = 1) = \mathbb{P}(C_k = 1 \mid E_k = 1)\mathbb{P}(E_k = 1) = \theta_k \, \text{rel}(p_k)$.

For (b), Definition 2.5.3(4) implies S_{k-1} is $\mathcal{F}_{k-1}$-measurable and depends on prior clicks/purchases. By Definition 2.5.3(3), $\mathbb{P}(E_k = 1 \mid \mathcal{F}_{k-1})$ depends on S_{k-1}, hence on past outcomes, which induces dependence across positions. □

Proposition 2.5.5 (Stopping Time Validity)

The stopping time τ from (2.8) is a valid stopping time with respect to the natural filtration $\mathcal{F}_k = \sigma(E_1, C_1, B_1, S_1, \ldots, E_k, C_k, B_k, S_k)$.

Proof. We verify that $\{\tau \leq k\} \in \mathcal{F}_k$ for all $k \geq 1$. The event $\{\tau \leq k\}$ is the union:

$$\{\tau \leq k\} = \bigcup_{j=1}^{k} \left(\{E_j = 0\} \cup \{S_j < \theta_{\text{abandon}}\} \cup \left\{ \sum_{\ell \leq j} B_\ell \geq n_{\max} \right\} \right).$$

Each component event at position $j \leq k$ is determined by $(E_1, \ldots, E_j, C_1, \ldots, C_j, B_1, \ldots, B_j, S_1, \ldots, S_j)$, all of which are $\mathcal{F}_k$-measurable for $j \leq k$. The union of $\mathcal{F}_k$-measurable events is $\mathcal{F}_k$-measurable. □

Remark 2.5.4 (Why this proof matters). The stopping time validity ensures that the session outcome $X_\tau = (C_1, \ldots, C_\tau, B_1, \ldots, B_\tau)$ is a well-defined random variable on the underlying probability space. This is essential for defining rewards (Chapter 1) and value functions (Chapter 3) rigorously. Without this guarantee, the "GMV of a session" would be mathematically undefined.

Code <-> Theory (Complete Mapping for Utility-Based Cascade)

Every term in Definition 2.5.3 maps directly to `BehaviorConfig` and `behavior.py`:

Theory Symbol	Code Reference	Default Value
α_{rel}	`BehaviorConfig.alpha_rel`	1.0
α_{price}	`BehaviorConfig.alpha_price`	0.8
α_{pl}	`BehaviorConfig.alpha_pl`	1.2
α_{cat}	`BehaviorConfig.alpha_cat`	0.6
σ_u	`BehaviorConfig.sigma_u`	0.8
$\text{pos\_bias}_k(q)$	`BehaviorConfig.pos_bias[query_type][k]`	see config
β_{exam}	hardcoded `0.2` in `behavior.py:91`	0.2
γ_{gain}	`BehaviorConfig.satisfaction_gain`	0.5
γ_{decay}	`BehaviorConfig.satisfaction_decay`	0.2
γ_{fatigue}	`BehaviorConfig.post_purchase_fatigue`	1.2
θ_{abandon}	`BehaviorConfig.abandonment_threshold`	-2.0
$n_{\max}$	`BehaviorConfig.max_purchases`	3

Implementation location: `zoosim/dynamics/behavior.py:67-118` (`simulate_session` function).

We have now formalized three click models with increasing realism: PBM (§2.5.1) for analytical tractability, DBN (§2.5.2) for cascade dynamics, and the Utility-Based Cascade (§2.5.4) for production simulation. The nesting property (Proposition 2.5.4) ensures that insights from the simpler models transfer to the richer one. Next, we turn to off-policy evaluation, where all three models serve as the outcome distribution $P(\cdot \mid x, a)$ in importance sampling.

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**.

Throughout this section, we write contexts as $x \sim \mathcal{D}$ for a distribution $\mathcal{D}$ on $\mathcal{X}$, propensities as $\pi_0(a \mid x)$ and $\pi_1(a \mid x)$, and we reserve ρ for importance weights (Radon-Nikodym derivatives) as in 2.3.5.

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.

2.6.2 Propensity Scores and Inverse Propensity Scoring (IPS)

To formally define the IPS estimator and prove its unbiasedness, we first state the regularity conditions required for off-policy evaluation.

Assumption 2.6.1 (OPE Probability Conditions).

For all $(x, a) \in \mathcal{X} \times \mathcal{A}$:

1. **Measurability**: $R(x, a, \omega)$ is measurable as a function of ω.
2. **Integrability**: $\mathbb{E}[|R(x, a, \omega)|] < \infty$ (finite first moment).
3. **Overlap (policy absolute continuity)**: For each context x, the evaluation policy $\pi_1(\cdot \mid x)$ is absolutely continuous with respect to the logging policy $\pi_0(\cdot \mid x)$, written $\pi_1(\cdot \mid x) \ll \pi_0(\cdot \mid x)$. In the finite-action case this is the support condition $\pi_1(a \mid x) > 0 \Rightarrow \pi_0(a \mid x) > 0$.

Conditions (1)–(2) ensure $Q(x, a) := \mathbb{E}[R(x, a, \omega) \mid x, a]$ is a well-defined finite expectation. Condition (3) is the **coverage** or **overlap** assumption: it guarantees that the importance weight is well-defined. In the finite-action case this weight is the ratio $\pi_1(a \mid x)/\pi_0(a \mid x)$ in (2.9); in general it is the Radon–Nikodym derivative $d\pi_1(\cdot \mid x)/d\pi_0(\cdot \mid x)$.

> **Remark 2.6.1a (Why overlap?).** Off-policy evaluation reweights logged actions: we estimate the value of π_1 using data collected under π_0. In the finite-action case, IPS uses the ratio $\pi_1(a \mid x)/\pi_0(a \mid x)$. For general action spaces, this ratio is replaced by the Radon–Nikodym derivative $d\pi_1(\cdot \mid x)/d\pi_0(\cdot \mid x)$. The overlap condition $\pi_1(\cdot \mid x) \ll \pi_0(\cdot \mid x)$ is exactly what guarantees that this derivative exists.
>
> **Remark 2.6.1b (Verification for our setting).** In the search ranking simulator (Chapters 4–5): condition (1) holds because rewards aggregate measurable click outcomes; condition (2) holds because rewards have **finite first moments**—prices are lognormal (finite moments), purchases per session are bounded by `BehaviorConfig.max_purchases`, and click counts are bounded by

`top_k`; condition (3) requires sufficient **exploration** in the logging policy—e.g., an ε-greedy policy with $\varepsilon > 0$ ensures all actions have positive probability, satisfying coverage.

Definition 2.6.1 (Propensity Score)

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 the conditional probability $\pi_0(a \mid x)$.

Definition 2.6.2 (Inverse Propensity Scoring Estimator)

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

$$\hat{V}_{\text{IPS}}(\pi_1) := \frac{1}{N} \sum_{i=1}^{N} \frac{\pi_1(a_i \mid x_i)}{\pi_0(a_i \mid x_i)} r_i. \tag{2.9}$$

Theorem 2.6.1 (Unbiasedness of IPS)

Under Assumption 2.6.1 (OPE Probability Conditions) and assuming **correct logging** (observed actions a_i are sampled from $\pi_0(\cdot \mid x_i)$), the IPS estimator is **unbiased**:

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

Proof.

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

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

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):

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

□

Remark 2.6.1 (Importance sampling mechanism). This proof is a finite-action instantiation of the Radon-Nikodym identity in 2.3.5. For each fixed context x, define $\mu(a) := \pi_0(a \mid x)$ and $\nu(a) := \pi_1(a \mid x)$ on $\mathcal{A}$. The weight $\rho(a \mid x) = d\nu/d\mu$ satisfies

$$\sum_{a\in\mathcal{A}} \rho(a \mid x)\, R(x,a)\, \mu(a) = \sum_{a\in\mathcal{A}} R(x,a)\, \nu(a),$$

which is exactly the algebra used in Steps 2-3.

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**.

Definition 2.6.3 (Clipped IPS)

For a cap $c > 0$, the **clipped IPS** estimator is

$$\hat{V}_{\text{clip}}(\pi_1) := \frac{1}{N}\sum_{i=1}^{N} \min\left\{c,\ \frac{\pi_1(a_i \mid x_i)}{\pi_0(a_i \mid x_i)}\right\} r_i. \tag{2.10}$$

Definition 2.6.4 (Self-normalized IPS, SNIPS)

The **SNIPS** estimator normalizes by the sum of weights:

$$\hat{V}_{\text{SNIPS}}(\pi_1) := \frac{\sum_{i=1}^{N} \frac{\pi_1(a_i|x_i)}{\pi_0(a_i|x_i)} r_i}{\sum_{i=1}^{N} \frac{\pi_1(a_i|x_i)}{\pi_0(a_i|x_i)}}. \tag{2.11}$$

Remark 2.6.5 (SNIPS properties). SNIPS reduces variance but loses unbiasedness; under mild regularity it is consistent as $N \to \infty$. Clipped IPS (2.10) trades bias for variance: for nonnegative rewards, clipping induces **negative bias** (the estimator systematically underestimates). Formal bias and consistency results, along with numerical experiments illustrating the bias–variance trade-off, live in **Chapter 9** (Off-Policy Evaluation), specifically 9.6.1 and Lab 9.5.

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): Let $w_\pi(p \mid x) > 0$ be a positive weight for product p under policy π in context x (for a real-valued score $s_\pi(p \mid x)$, a standard choice is $w_\pi(p \mid x) = \exp(s_\pi(p \mid x))$). Approximate the ranking distribution via sequential sampling. Let R_k be the set of remaining items after positions $1, \dots, k-1$ have been chosen: $R_k := \{p : p \notin \{p_1, \dots, p_{k-1}\}\}$. Then

$$\pi(p_k \mid x, p_1, \dots, p_{k-1}) = \frac{w_\pi(p_k \mid x)}{\sum_{p \in R_k} w_\pi(p \mid x)}. \tag{2.12}$$

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

$$\pi(a \mid x) = \prod_{k=1}^{M} \frac{w_\pi(p_k \mid x)}{\sum_{p \in R_k} w_\pi(p \mid x)}. \tag{2.13}$$

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

$$\pi_0^{(K)}(a \mid x) := \prod_{k=1}^{K} \frac{w_{\pi_0}(p_k \mid x)}{\sum_{j=k}^{M} w_{\pi_0}(p_j \mid x)}.$$

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.

Counterexample 2.6.1 (Factorization bias under item-position weights). Consider two items A, B with logging scores $s_0(A) = 2$, $s_0(B) = 1$ and target scores $s_1(A) = 1$, $s_1(B) = 2$. Under Plackett–Luce,

$$\mu((A,B)) = \tfrac{2}{3}, \quad \mu((B,A)) = \tfrac{1}{3}, \qquad \pi((A,B)) = \tfrac{1}{3}, \quad \pi((B,A)) = \tfrac{2}{3}.$$

Let reward R be 1 if the top item is A and 0 otherwise. The true value is $V(\pi) = \frac{1}{3}$. From these ranking probabilities, we derive item-position marginals:

$$\mu(A@1) = \tfrac{2}{3},\ \mu(B@2) = \tfrac{2}{3},\ \mu(B@1) = \tfrac{1}{3},\ \mu(A@2) = \tfrac{1}{3}$$

$$\pi(A@1) = \tfrac{1}{3},\ \pi(B@2) = \tfrac{1}{3},\ \pi(B@1) = \tfrac{2}{3},\ \pi(A@2) = \tfrac{2}{3}$$

The **item-position factorization** that uses per-position marginals yields

$$\tilde{w}(A,B) = \frac{\pi(A@1)}{\mu(A@1)} \cdot \frac{\pi(B@2)}{\mu(B@2)} = \frac{1/3}{2/3} \cdot \frac{1/3}{2/3} = \tfrac{1}{4}, \quad \tilde{w}(B,A) = \frac{2/3}{1/3} \cdot \frac{2/3}{1/3} = 4.$$

Hence $\mathbb{E}_\mu[\tilde{w}R] = \frac{2}{3} \cdot \frac{1}{4} \cdot 1 + \frac{1}{3} \cdot 4 \cdot 0 = \frac{1}{6} \neq V(\pi) = \frac{1}{3}$. The factorization produces **biased IPS** because item-position marginals ignore the correlation structure of full rankings. List-level propensities (product of conditionals, (2.13)) avoid this pitfall.

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 `zoosim/evaluation/ope.py`.

2.7 Computational Illustrations

We verify the theory numerically using simple Python experiments.

2.7.1 Simulating PBM and DBN Click Models

We 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?")
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)
```

```python
    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
                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

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
# Note: tau is 0-based index; add 1 for position (rank)
print(f"\n\nMean stopping position (1-based): {tau_dbn.mean() + 1:.2f}")
print(f"% sessions satisfied at position 1: {(tau_dbn == 0).mean() * 100:.1f}%")
print(f"% sessions exhausting list without satisfaction: {(tau_dbn == M).mean() * 100:.1f}%'

# Output:
# Mean stopping position (1-based): 3.34
# % sessions satisfied at position 1: 48.6%
# % sessions exhausting list without satisfaction: 5.2%
```

Key observations: 1. **PBM verification**: Empirical CTR matches 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. **Early satisfaction**: ~50% satisfied at position 1; only ~5% exhaust list without satisfaction

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

Remark 2.7.1 (Confidence intervals). For Bernoulli CTR estimates with n sessions, a 95% normal approximation interval is $\hat{p} \pm 1.96\sqrt{\hat{p}(1-\hat{p})/n}$. With $n = 50{,}000$, the tolerance < 0.01 used above lies within the corresponding confidence intervals.

2.7.2 Verifying IPS Unbiasedness

We 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-2.9]
    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):
    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 (une

# 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.

Code <-> Env/Reward (session step and aggregation)

The end-to-end simulator routes theory to code: - Env step calls behavior: `zoosim/envs/search_env.py:93-100` (`behavior.simulate_session`) - Reward aggregation per (**??**): `zoosim/dynamics/reward.py:60-65` - Env returns ranking, clicks, buys: `zoosim/envs/search_env.py:113-120`

2.7.3 Verifying the Tower Property Numerically

We illustrate the Tower Property 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.

```
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 $\mathbb{E}[Z \mid \mathcal{G}]$.

Code <-> Theory (Tower Property)

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

2.8 Application Bridge to RL

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

Before diving into MDPs, we establish a key integrability result that ensures reward expectations are well-defined.

Proposition 2.8.1 (Score Integrability)

Under the standard Borel assumptions (see §2.1 assumptions box), the base relevance score function $s : \mathcal{Q} \times \mathcal{P} \to \mathbb{R}$ satisfies:

1. **Measurability**: s is $(\mathcal{B}(\mathcal{Q}) \otimes \mathcal{B}(\mathcal{P}), \mathcal{B}(\mathbb{R}))$-measurable
2. **Square-integrability**: Under the simulator-induced distribution, $s \in L^2$

Proof.

Step 1 (Score decomposition). The simulator's relevance score decomposes as

$$s(q, p) = s_{\text{sem}}(q, p) + s_{\text{lex}}(q, p) + \varepsilon,$$

where s_{sem} is the cosine similarity between query and product embeddings, $s_{\text{lex}} = \log(1 + \text{overlap}(q, p))$ is the lexical overlap component, and $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ is independent observation noise.

Step 2 (Measurability). The cosine similarity $s_{\text{sem}} : \mathbb{R}^d \times \mathbb{R}^d \to [-1, 1]$ is continuous (hence Borel measurable) away from the origin. The logarithm $\log : (0, \infty) \to \mathbb{R}$ is continuous. Compositions and sums of Borel-measurable functions are Borel measurable ([@folland:real_analysis:1999, Proposition 2.6]). The noise ε is measurable as a random variable by construction. Thus s is $(\mathcal{B}(\mathcal{Q}) \otimes \mathcal{B}(\mathcal{P}) \otimes \mathcal{B}(\mathbb{R}), \mathcal{B}(\mathbb{R}))$-measurable.

Step 3 (Finite second moments). We verify each component:

- *Semantic component*: $|s_{\text{sem}}(q, p)| \leq 1$, so $s_{\text{sem}}^2 \leq 1$.
- *Lexical component*: The overlap count is bounded by token-set sizes: $\text{overlap}(q, p) \leq \min(|q|, |p|) \leq M$ where M is the maximum query/product token count in the simulator. Thus $s_{\text{lex}}^2 \leq (\log(1 + M))^2 < \infty$.
- *Noise component*: $\mathbb{E}[\varepsilon^2] = \sigma^2 < \infty$ by assumption.

Step 4 (Square-integrability of sum). By independence of ε from (q, p) and the elementary inequality $(a + b + c)^2 \leq 3(a^2 + b^2 + c^2)$,

$$\mathbb{E}[s(Q, P)^2] \leq 3\big(\mathbb{E}[s_{\text{sem}}^2] + \mathbb{E}[s_{\text{lex}}^2] + \mathbb{E}[\varepsilon^2]\big) \leq 3\big(1 + (\log(1 + M))^2 + \sigma^2\big) < \infty.$$

Thus $s \in L^2$ under the simulator-induced distribution. □

Remark 2.8.1a (Boundedness not required). The OPE machinery (Theorem 2.6.1) requires only integrability, not boundedness. Our scores are square-integrable but **not** bounded to $[0, 1]$—the Gaussian noise component is unbounded. This is typical of realistic relevance models where measurement noise and model uncertainty introduce unbounded perturbations.

Consequence for OPE (Theorem 2.6.1): The integrability assumption in Theorem 2.6.1 is satisfied when rewards have **finite first moments**. Since $R(x, a, \omega)$ aggregates GMV (lognormal prices with finite moments), CM2 (bounded margin rates), and clicks (bounded by `top_k`)—all with finite expectations—the IPS estimator is well-defined.

Code <-> Theory (Score Distribution)

Lab 2.2 verifies Proposition 2.8.1 empirically. The implementation in `zoosim/ranking/relevance.py` combines:

- Cosine similarity (bounded $[-1, 1]$)
- Lexical overlap $\log(1 + \text{overlap})$ (bounded by token-set sizes)
- Gaussian noise (unbounded but integrable)

Scores are NOT constrained to $[0, 1]$; empirically they cluster near 0 with std ~ 0.2.

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) - $P(\cdot \mid s, a)$: Markov transition kernel on $\mathcal{S}$ (for each (s, a), $P(\cdot \mid s, a)$ is a probability measure on $\mathcal{S}$, and $(s, a) \mapsto P(B \mid s, a)$ is measurable for each measurable $B \subseteq \mathcal{S}$) - $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$: measurable one-step reward function (the realized reward random variables are $R_t := R(S_t, A_t)$ on the induced trajectory space) - $\gamma \in [0, 1)$: 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 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 2.8.1b (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.

Proposition 2.8.2 (Bellman measurability and contraction) .

Assume standard Borel state/action spaces, bounded measurable rewards $r(s,a)$, measurable Markov kernel $P(\cdot \mid s,a)$, measurable policy kernel $\pi(\cdot \mid s)$, and discount $0 \le \gamma < 1$. Then on $(B_b(\mathcal{S}), \|\cdot\|_\infty)$, - the policy evaluation operator

$$(\mathcal{T}^\pi V)(s) := \int_{\mathcal{A}} r(s,a)\,\pi(da \mid s) + \gamma \int_{\mathcal{A}}\int_{\mathcal{S}} V(s')\,P(ds' \mid s,a)\,\pi(da \mid s) \tag{2.14}$$

maps bounded measurable functions to bounded measurable functions and satisfies $\|\mathcal{T}^\pi V - \mathcal{T}^\pi W\|_\infty \le \gamma\|V-W\|_\infty$; - the control operator

$$(\mathcal{T}V)(s) := \sup_{a\in\mathcal{A}}\Big\{r(s,a) + \gamma\int_{\mathcal{S}} V(s')\,P(ds' \mid s,a)\Big\}$$

is a γ-contraction under the same boundedness conditions; measurability of $\mathcal{T}V$ may require additional topological assumptions (e.g., compact $\mathcal{A}$, upper semicontinuity) or a measurable selection theorem.

Proof sketch. Measurability is preserved under integration against Markov kernels; boundedness follows from bounded r and V. The contraction bound follows from the triangle inequality and linearity of the integral. □

Remark 2.8.2 (Control operator) . For the control operator

$$(\mathcal{T}V)(s) := \sup_{a\in\mathcal{A}}\Big\{r(s,a) + \gamma\int_{\mathcal{S}} V(s')\,P(ds' \mid s,a)\Big\},$$

measurability of $\mathcal{T}V$ can require additional topological assumptions (e.g., compact $\mathcal{A}$ and upper semicontinuity) or application of a measurable selection theorem. Contraction still holds under boundedness and $0 \le \gamma < 1$. The measurable selection theorem needed for the control operator appears in **§2.8.2 (Advanced: Measurable Selection and Optimal Policies)** below.

2.8.2 (Advanced) Measurable Selection and Optimal Policies

This section is optional on a first reading. It addresses the topological fine print that ensures optimal policies $\pi^*(s) = \arg\max_a Q(s,a)$ are well-defined measurable functions in continuous state and action spaces. Readers primarily interested in finite action spaces (Chapters 6, 8) can safely skip this and return when studying continuous actions (Chapter 7).

In Remark 2.8.2 above, we noted that the control Bellman operator

$$(\mathcal{T}V)(s) = \sup_{a \in \mathcal{A}} \left\{ r(s,a) + \gamma \int_{\mathcal{S}} V(s') P(ds' \mid s,a) \right\}$$

requires additional care to ensure measurability of $\mathcal{T}V$. The issue is subtle but fundamental to continuous-space RL.

The problem. For each state s, the Bellman optimality equation requires finding $a^*(s) = \arg\max_a Q(s,a)$. But knowing that a maximum *exists* at each s (e.g., by compactness and continuity) doesn't guarantee that the mapping $s \mapsto a^*(s)$ is *measurable*—and if it's not measurable, the "optimal policy" cannot be evaluated as a random variable. We couldn't take expectations like $\mathbb{E}[R(S, \pi^*(S))]$ because π^* would be ill-defined measure-theoretically.

Why this is subtle. The supremum of measurable functions is measurable—this is a standard result in measure theory. But the **argmax** (the action achieving the supremum) need not be measurable. Geometrically, the set of maximizers

$$A^*(s) = \{a \in \mathcal{A} : Q(s,a) = \sup_{a' \in \mathcal{A}} Q(s,a')\}$$

can vary wildly with s in continuous spaces. Selecting one element from each set $A^*(s)$ in a measurable way is non-trivial—this is an Axiom of Choice problem constrained by measurability requirements. Without structure, pathological examples exist where no measurable selector is possible.

The solution. The following theorem packages the conditions under which measurable optimal policies exist:

Theorem 2.8.3 (Kuratowski–Ryll–Nardzewski Selection, specialized to RL).

Let $\mathcal{S}$ be a standard Borel space and $\mathcal{A}$ a compact metric space. Suppose $Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ satisfies:

(A1) Joint measurability: Q is Borel measurable in (s,a)

(A2) Upper semicontinuity: For each fixed $s \in \mathcal{S}$, the map $a \mapsto Q(s,a)$ is upper semicontinuous

Then there exists a Borel measurable function $\pi^* : \mathcal{S} \to \mathcal{A}$ such that

$$Q(s, \pi^*(s)) = \sup_{a \in \mathcal{A}} Q(s,a) \quad \text{for all } s \in \mathcal{S}.$$

That is, a **measurable optimal selector** (greedy policy) exists.

Proof sketch. Assumption (A2) plus compactness of $\mathcal{A}$ ensure the supremum is attained at each s (Weierstrass theorem). The challenge is proving that some selector is measurable. Under standard Borel state spaces and compact metric action spaces, the Kuratowski–Ryll–Nardzewski measurable selection theorem

([@kuratowski:selectors:1965]) guarantees that every non-empty-valued, closed-valued measurable correspondence admits a measurable selector. The set-valued map $s \mapsto A^*(s)$ (maximizers) is closed-valued by upper semicontinuity and measurable by joint measurability of Q. Applying the theorem yields a measurable π^*. See [@kechris:classical_dsp:1995, §36] for the full descriptive set theory machinery. $\square$

What this guarantees for RL. This theorem ensures:

1. **Bellman optimality operator is well-defined:** The pointwise maximum $(\mathcal{T}V)(s) = \max_a\{r(s,a) + \gamma\mathbb{E}[V(S') \mid s,a]\}$ produces a measurable function $\mathcal{T}V : \mathcal{S} \to \mathbb{R}$ when V is measurable.
2. **Optimal policies are random variables:** The greedy policy $\pi^*(s) \in \arg\max_a Q(s,a)$ is a measurable function, so we can evaluate expectations like $\mathbb{E}_{S\sim\rho_0}[R(S,\pi^*(S))]$ when S is drawn from an initial state distribution ρ_0.
3. **Value iteration convergence machinery works:** Chapter 3 proves that iterating $V_{n+1} = \mathcal{T}V_n$ converges to the unique fixed point V^*. Measurability of $\mathcal{T}$ at each step is essential for this operator-theoretic argument.

When this theorem matters in our book:

- **Finite action spaces:** Trivially satisfied. If $\mathcal{A} = \{a_1, \dots, a_M\}$ is finite, compactness is automatic and argmax is trivially measurable (just pick the first maximizer in some fixed ordering). No sophisticated machinery needed.
- **Discrete template bandits (Chapter 6):** Our template library has $|\mathcal{A}| = 25$ templates. Theorem 2.8.3 is overkill—measurability is immediate.
- **Continuous boost actions (Chapter 7):** When we learn $Q(x,a)$ for $a \in [-a_{\max}, a_{\max}]^K$ (continuous action space), this theorem becomes essential. Neural Q-networks approximate Q as $Q_\theta(x,a)$; upper semicontinuity (A2) may fail in practice (neural nets are not inherently u.s.c.), requiring care in implementation.
- **General state/action spaces:** In extensions to continuous state representations (e.g., learned embeddings $x \in \mathbb{R}^d$), the standard Borel assumption on $\mathcal{S}$ becomes critical. Pathological spaces exist where measurable selection fails without these topological conditions.

Practical takeaway. In the finite-action and compact-action settings emphasized early in this book, measurability of greedy selectors is immediate; the selection theorem becomes essential primarily when passing to genuinely continuous action spaces. On a first reading, we may treat the set-theoretic details as background and focus on the consequences stated above.

References: The original Kuratowski–Ryll–Nardzewski theorem appears in [@kuratowski:selectors:1965]. For textbook treatments: [@kechris:classical_dsp:1995, §36] provides the definitive descriptive set theory perspective; [@bertsekas:stochastic_oc:1996, Chapter 7] develops the RL-specific machinery and verifies standard Borel conditions for typical RL state/action spaces.

2.8.3 Click Models as Contextual Bandits

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

Setup: - Context $x \sim \mathcal{D}$ (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, (**??**))

Goal: Learn policy π^* maximizing

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

Click models provide $P(\cdot \mid x, a)$: - PBM (Section 2.5.1): $\omega = (E_1, C_1, \dots, E_M, C_M)$ with independent examination/click - DBN (Section 2.5.2): $\omega = (E_1, C_1, S_1, \dots)$ 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 (2.9) - 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 10** (Guardrails): Production constraints—CM2 floors, ΔRank@k stability, safe fallback (CMDP theory in §3.6) - **Chapter 9** (Off-Policy Evaluation): Production IPS, SNIPS, doubly robust estimators

All rely on the probability foundations built in this chapter.

2.8.4 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

---

2.9 Production Checklist

Production Checklist (Chapter 2)

Configuration alignment:
The simulator implements the **Utility-Based Cascade Model** (§2.5.4). Under a parameter specialization it reproduces PBM's marginal factorization (Proposition 2.5.4), and in general it exhibits cascade-style dependence through an internal state.

- **Position bias**: `BehaviorConfig.pos_bias` in `zoosim/core/config.py:180-186` — dictionary mapping query types to position bias vectors (PBM-like behavior)
- **Satisfaction dynamics**: `BehaviorConfig.satisfaction_decay` and `satisfaction_gain` in `zoosim/core/config.py:175-176` — state-driven cascade dependence (DBN-like mechanism)
- **Abandonment threshold**: `BehaviorConfig.abandonment_threshold` in `zoosim/core/config.py:177` — session termination condition
- **Seeds**: `SimulatorConfig.seed` in `zoosim/core/config.py:252` for reproducible click patterns

Implementation modules:

- `zoosim/dynamics/behavior.py`: Implements cascade session simulation via `simulate_session()` — combines PBM position bias with DBN-style satisfaction/abandonment dynamics
- `zoosim/core/config.py`: `BehaviorConfig` dataclass with utility weights (`alpha_*`), satisfaction parameters, and position bias vectors
- `zoosim/evaluation/ope.py` (Chapter 9): Implements IPS, SNIPS, PDIS, and DR estimators from Definitions 2.6.1–2.6.2

Tests:

- `tests/ch09/test_ope.py`: Verifies OPE estimator behavior
- `tests/ch02/test_behavior.py`: Verifies position bias monotonicity (click rates decrease with position); does NOT verify exact PBM/DBN equation matches
- `tests/ch02/test_segment_sampling.py`: Verifies segment frequencies converge to configured probabilities

Score distribution (Lab 2.2):

- Lab 2.2 validates integrability; scores are NOT bounded to $[0, 1]$
- Empirically, scores cluster near 0 with std ~ 0.2

Reward moments:

- GMV/CM2 have finite moments (lognormal prices); not bounded
- Click counts bounded by `top_k`

Assertions:

- Check $0 \leq \theta_k \leq 1$ for all position bias values
- Check positivity assumption $\pi_0(a \mid x) > 0$ when computing IPS weights

---

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 $\mathrm{rel}(p_3) = 0.6$ 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: - $\mathrm{rel}(p_1) = 0.8$, $s(p_1) = 0.5$ - $\mathrm{rel}(p_2) = 0.6$, $s(p_2) = 0.7$ - $\mathrm{rel}(p_3) = 0.9$, $s(p_3) = 0.9$

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 $\hat{V}_{\mathrm{IPS}}(\pi_1)$ 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 $D_{\mathrm{KL}}(\pi_1 \| \pi_0)$). Verify that variance increases as policies diverge.

Labs

- Lab 2.1 — Segment Mix Sanity Check: sample thousands of users via `zoosim.users` and verify empirical frequencies converge to the segment distribution $\mathbf{p}_{\text{seg}}$ from 2.2.6.
- Lab 2.2 — Query Measure and Base Score Integration: connect the PBM/DBN derivations to simulator base scores by logging statistics from `zoosim.queries` and `zoosim.relevance`.
- Lab 2.3 — Textbook Click Model Verification: verify PBM and DBN toy simulators match the closed-form predictions from (2.1) and (2.3).
- Lab 2.4 — Nesting Verification ([PROP-2.5.4]): show that the Utility-Based Cascade reduces to PBM under the parameter specialization in 2.5.4.
- Lab 2.5 — Utility-Based Cascade Dynamics ([DEF-2.5.3]): empirically validate position decay, satisfaction dynamics, and stopping conditions from [EQ-2.4]–[EQ-2.8].

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: $\mathbb{E}[R \mid W]$ is a $\sigma(W)$-measurable conditional expectation, and (under standard Borel assumptions) $\mathbb{E}[R \mid W = w] = \int r \, d\mathbb{P}(R \in dr \mid W = w)$ is a regular conditional expectation. - **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 under positivity and integrability; clipped IPS incurs negative bias and SNIPS trades bias for variance.

Next chapter: We develop stochastic processes, Markov chains, Bellman operators, and contraction mappings, and we prove value iteration convergence via the Banach fixed-point theorem.

Central lesson: Reinforcement learning is applied probability theory: algorithms are expectations, policies are conditional distributions, and convergence proofs rely on measure-theoretic limit theorems. This chapter supplies the hypotheses required for those results.

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 `docs/references.bib`.