A/B Testing & Iterative Improvement

Week 10: Closing the ML Innovation Loop

Machine Learning for Smarter Innovation

BSc-Level Course

Overview

The Iteration Advantage: Why Speed Wins

The Winners Iterate Fast

Spotify:

- Tens of thousands of experiments annually
- Test every feature before launch
- 30% faster iteration than competitors
- Result: 500M+ users

Netflix:

- AI recommendation system saves $1B annually
- A/B testing validates all improvements
- 80% engagement from personalization
- Tests 250+ variations simultaneously

Amazon:

- Tests every UI change
- 35% revenue from recommendations (McKinsey)
- "Our success is a function of experiments"
- Iterates daily, not quarterly

The Losers Deploy and Hope

Traditional Approach:

- Build for 6 months
- Launch to 100% users
- Hope it works
- No measurement
- No iteration

The Iteration Gap

Industry Reality:

- 87-90% of ML projects never reach production
- Most that deploy: No iteration, no measurement
- Majority lack A/B testing infrastructure
- Average: 2-3 experiments per year vs 50+

Week 10 Mission

Transform you from "deploy and hope" to "test and learn" practitioners who ship 50+ experiments annually.

Experimental velocity correlates with innovation output - systematic testing infrastructure accelerates discovery rates

Why Deployed Models Decay Without Iteration

The Decay Curve

Typical Model Degradation:

- Month 0: 90% accuracy (deployment)
- Month 3: 87% accuracy (3% drop)
- Month 6: 82% accuracy (9% drop)
- Month 12: 75% accuracy (17% drop)
- Month 18: 68% accuracy (24% drop)

Performance drops 15-30% per year without updates

Why This Happens:

- User behavior changes
- Competitor adaptations
- Market dynamics shift
- Seasonal patterns evolve
- New product features launched

Root Causes

1. Concept Drift

- User preferences change
- $P(Y|X)$ relationship shifts
- Training data becomes stale

2. Data Distribution Shift

- New user demographics
- $P(X)$ changes over time
- Features no longer predictive

3. Competitive Response

- Rivals launch better features
- Users have higher expectations
- Bar constantly rising

The Solution

Continuous iteration through A/B testing:

- Weekly model updates
- Feature experiments
- Algorithm improvements

From "Deploy and Hope" to "Test and Learn"

Old Mindset: Waterfall

Process:

1. Spend 6 months building
2. Launch to all users
3. Cross fingers
4. Wait for complaints
5. Emergency fix if broken
6. Repeat in 12 months

Problems:

- High risk (all eggs in one basket)
- Slow learning (feedback after 6 months)
- No data (guessing what works)
- Expensive failures
- Competitor advantage grows

Success rate: 10-20%

New Mindset: Scientific Method

Process:

1. Form hypothesis ("X will improve Y")
2. Design experiment (A/B test)
3. Calculate sample size (power analysis)
4. Deploy to 5% users (low risk)
5. Measure results (data-driven)
6. Ship winners, kill losers
7. Repeat weekly

Benefits:

- Low risk (gradual rollout)
- Fast learning (results in days)
- Data-driven (know what works)
- Compound gains (many small wins)
- Competitive edge (10*imes* iteration speed)

Success rate: 30-40%

Key insight: Most experiments "fail" (don't beat control), but you learn fast and iterate

A/B Testing Fundamentals: The Gold Standard

What is A/B Testing?

Definition: Randomized controlled trial comparing two versions to determine which performs better.

Core Components:

1. **Control Group (A):** Current system (baseline)
2. **Treatment Group (B):** New variant (challenger)
3. **Random Assignment:** Users split 50/50
4. **Metric:** What you're measuring (CTR, revenue)
5. **Statistical Test:** Determine if difference is real

Example: Recommendation Algorithm

- Control: Popularity-based (5% CTR)
- Treatment: ML collaborative filtering
- Metric: Click-through rate
- Users: 50,000 per group
- Duration: 2 weeks
- Result: Treatment wins (6.2% CTR)

Why Randomization Matters

Without randomization:

- Selection bias (power users in treatment)
- Confounding variables (time of day, device)
- Can't isolate causal effect
- Results are unreliable

With randomization:

- Groups are statistically equivalent
- Only difference is the treatment
- Causal interpretation valid
- Results are trustworthy

Key Requirements

- **Sample size:** Large enough for statistical power
- **Duration:** Long enough to capture true behavior
- **Randomization:** Truly random assignment
- **Isolation:** No spillover between groups
- **Pre-registration:** Hypothesis before experiment

A/B testing applies scientific method to product development - controlled experiments enable evidence-based decision-making

Real Success Stories: Data-Driven Wins

Booking.com

Experiment: Scarcity messaging

Hypothesis:
"Only 2 rooms left!" increases urgency and conversions

Test:

- Control: No scarcity message
- Treatment: Real-time room count
- Metric: Conversion rate
- Users: 100K per group

Result:

- 25% conversion lift
- $50M annual revenue impact
- Rolled out globally

Lesson: Small UX changes, massive impact when tested rigorously

Obama 2012 Campaign

Experiment: Email subject lines

Challenge:
Raise $500M through email donations

Test:

- 20 subject line variants
- A/B/C/D.../T testing
- Metric: Donation rate
- Recipients: 13M per variant

Result:

- Winner: "Hey" (simplest)
- 70% higher open rate
- $500K extra per email
- 500+ experiments total

Lesson: Intuition fails. Data wins. Test everything.

Google Search Ads

Experiment: 41 shades of blue

Question:
Which blue color for ad links maximizes clicks?

Test:

- 41 blue color variants
- Metric: Click-through rate
- Traffic: Billions of searches
- Duration: 3 weeks

Result:

- Optimal blue found
- 1% CTR improvement
- $200M annual revenue
- Criticized but data-driven

Lesson: At scale, tiny improvements = huge value. Test obsessively.

Empirical evidence supersedes expert intuition - measured outcomes reveal truths hidden from qualitative assessment

Real Failure Examples: The Cost of Not Testing

Knight Capital: $440M Loss

What Happened:

- August 1, 2012: Deploy new trading algorithm
- No A/B test, no gradual rollout
- Bug: Bought high, sold low repeatedly
- Duration: 45 minutes
- Loss: $440 million
- Outcome: Company bankrupt

What Should Have Been Done:

- Shadow mode testing (parallel run)
- Canary deployment (1% traffic first)
- Kill switch (automatic rollback)
- Guardrail metrics (loss limits)

Lesson: High-stakes deployments demand rigorous testing

Microsoft Tay Chatbot

What Happened:

- March 2016: Launch AI chatbot on Twitter
- No gradual rollout, no guardrails
- Users exploited vulnerabilities
- Within 16 hours: Racist, offensive tweets
- Outcome: Shut down, PR disaster

What Should Have Been Done:

- Beta test with controlled users
- Content moderation safeguards
- Incremental rollout (100 $\rightarrow$ 1K $\rightarrow$ 10K users)
- Real-time monitoring and kill switch

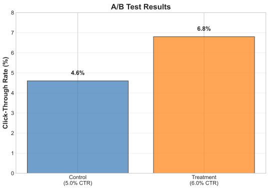
Lesson: AI in the wild requires safety testing

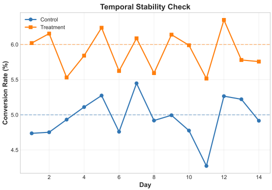
Common Thread

Both failures share: (1) No gradual rollout, (2) No guardrails, (3) No monitoring, (4) No rollback plan

Experimentation prevents catastrophic failures - systematic testing identifies risks before they manifest at scale

The Experimentation Pyramid: Three Levels





Level 1: Speed

Ship fast, iterate faster

- 50+ experiments per year
- 1-2 week experiment duration
- Automated analysis pipelines
- Low friction deployment

Level 2: Safety

Guardrails prevent disasters

- Revenue floor (don't lose money)
- Latency ceiling (stay fast)
- Error rate limits (don't break)
- Automatic kill switches

Level 3: Learning

Extract insights, compound gains

- Why did it win/lose?
- What worked for which users?
- How to generalize learnings?
- Build institutional knowledge

When to A/B Test ML Models

Always A/B Test

1. New Model Deployment

- Replacing existing model
- High risk of regression
- Need causal validation

2. Algorithm Changes

- Random Forest

→ XGBoost

- Content-based

→ collaborative filtering

- Offline metrics don't predict online

3. Feature Updates

- Add new features
- Remove stale features
- Feature engineering changes

4. Hyperparameter Tuning

- Learning rate changes
- Regularization adjustments

Sometimes A/B Test

6. UI/UX Changes

- Model recommendations display
- Explanation text
- Confidence thresholds shown to users

7. Business Logic

- Ranking vs filtering
- Diversity vs relevance
- Personalization intensity

Don't Bother A/B Testing

8. Bug Fixes

- Obvious correctness issues
- Fix immediately, no test needed

9. Infrastructure Changes

- Same model, faster serving
- No user-facing behavior change

Rule of Thumb

If change affects user experience or business metrics → A/B test it

Week 10 Learning Objectives

Technical Skills

By end of week, you will:

1. **Design** statistically rigorous A/B tests for ML models
2. **Calculate** sample sizes and experiment duration (power analysis)
3. **Implement** experiments in Python (scipy, PyMC3, bandits)
4. **Interpret** p-values, confidence intervals, effect sizes correctly
5. **Apply** Bayesian A/B testing for faster decisions
6. **Use** multi-armed bandits for real-time optimization
7. **Identify** and avoid common pitfalls (peeking, multiple testing)
8. **Build** experiment monitoring dashboards

Strategic Skills

You will master:

- Formulating testable hypotheses
- Choosing primary and guardrail metrics
- Balancing speed and statistical rigor
- Communicating results to stakeholders
- Making confident deployment decisions
- Building experimentation culture
- Measuring long-term impact
- Iterating systematically

Professional Outcomes

- Design and run 50+ experiments annually
- Avoid catastrophic deployment failures
- Iterate 10*imes* faster than competitors
- Build data-driven product culture
- Become trusted voice in product decisions

Validation establishes baseline quality while experimentation drives continuous improvement - measurement enables optimization

Foundation Summary: Why Iteration Wins

Key Concepts

1. Iteration Advantage

- Winners (Spotify, Netflix) ship 1000+ experiments/year
- Losers deploy once and hope
- Speed of learning = competitive edge

2. Model Decay

- Performance drops 15-30% annually without updates
- Concept drift, distribution shift, competition
- Solution: Continuous iteration

3. Experimentation Mindset

- From "deploy and hope" to "test and learn"
- Scientific method: Hypothesis

→ Test

→ Learn

- Most experiments "fail" but learning compounds

4. A/B Testing Fundamentals

- Randomized controlled trials
- Control vs treatment

Real-World Lessons

Success Stories:

- Booking.com: 25% conversion lift from scarcity
- Obama 2012: \$500K per optimized email
- Google: \$200M from testing blue shades

Failure Examples:

- Knight Capital: \$440M loss (no testing)
- Microsoft Tay: Shut down in 16 hours
- Lesson: Gradual rollout + guardrails essential

The Experimentation Pyramid

- **Speed:** Ship fast, iterate faster
- **Safety:** Guardrails prevent disasters
- **Learning:** Extract insights, compound gains

Next: Learn statistical techniques for rigorous A/B testing

Hypothesis Formulation: The Starting Point

Anatomy of a Good Hypothesis

SMART Framework:

- **S**pecific: Clearly defined change
- **M**easurable: Quantifiable outcome
- **A**ctionable: You can implement it
- **R**elevant: Aligns with business goals
- **T**ime-bound: Experiment duration set

Good Example

Hypothesis: Switching from content-based to collaborative filtering recommendations will increase click-through rate by at least 1 percentage point (from 5% to 6%) over a 2-week test period with 100,000 users.

Why good:

- Specific change (CF algorithm)
- Measurable outcome (CTR, 1 pp)
- Actionable (can deploy CF)
- Relevant (engagement goal)
- Time-bound (2 weeks, 100K users)

Bad Examples

Vague: "New algorithm will be better"

- Not specific (which algorithm?)
- Not measurable (better how?)
- Not time-bound

Unmeasurable: "Users will like recommendations more"

- "Like" not quantified
- No success criterion

Too ambitious: "Will 10*imes* revenue"

- Unrealistic expectation
- Sets up for disappointment

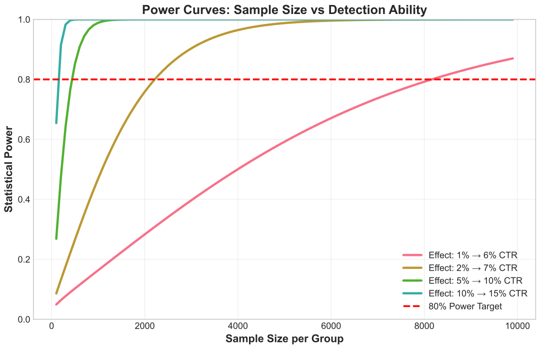
Null vs Alternative Hypothesis

Null (H_0): No difference between control and treatment
$CTR_{treatment} = CTR_{control}$

Alternative (H_1): Treatment is better
$CTR_{treatment} > CTR_{control}$ (one-tailed)
$CTR_{treatment} \neq CTR_{control}$ (two-tailed)

Hypothesis pre-registration prevents post-hoc rationalization - documenting predictions before observing results ensures scientific integrity

Sample Size Calculation: How Many Users?



Key Parameters

1. **Significance Level** (α)
   - Type I error rate (false positive)
   - Standard: 0.05 (5%)
   - Probability of detecting difference when none exists
2. **Statistical Power** (1 - β)
   - Type II error rate (false negative)
   - Standard: 0.80 (80%)
   - Probability of detecting true difference
3. **Minimum Detectable Effect (MDE)**
   - Smallest difference you care about
   - Example: 5%

→ 6% CTR (1 pp absolute, 20% relative)

- Smaller MDE needs larger sample

4. **Baseline Metric**
   - Current performance level
   - Variance affects sample size

Sample size determines statistical power - insufficient observations lead to inconclusive results regardless of analytical rigor

Randomization Strategies: More Than Just Coin Flips

Simple Randomization

Method: Coin flip per user (50/50)

Pros:

- Easy to implement
- Unbiased in expectation
- Good for large samples

Cons:

- Group sizes may differ
- Imbalanced covariates possible

Stratified Randomization

Method: Balance within strata (device, region)

Example:

- Split iOS users 50/50
- Split Android users 50/50
- Ensures device balance

Benefits:

- Reduces variance
- Controls confounders

Blocked Randomization

Method: Randomize within fixed blocks

Example:

- Block 1: AABBA (2A, 2B)
- Block 2: BAABB (2A, 2B)
- Guarantees equal sizes

Use case: Sequential enrollment

Clustered Randomization

Method: Randomize groups, not individuals

Example:

- User sessions (not individual actions)
- Geographic regions (not cities)
- Social networks (friends together)

Why:

- Prevent contamination
- Network effects
- Administrative ease

Drawback: Larger sample size needed

A/B Testing Math: Statistical Tests

Z-Test for Proportions

When: Binary outcome (click/no click)

Formula:

$$z = \frac{p_B - p_A}{\sqrt{p(1-p)(\frac{1}{n_A} + \frac{1}{n_B})}}$$

where $p = \frac{n_A p_A + n_B p_B}{n_A + n_B}$ (pooled proportion)

Example:

- Control: 500/10,000 clicked (5%)
- Treatment: 620/10,000 clicked (6.2%)
- $z = 4.72$, $p < 0.001$
- Conclusion: Significant improvement

T-Test for Means

When: Continuous outcome (revenue, time)

Formula:

$$t = \frac{\bar{x}_B - \bar{x}_A}{\sqrt{\frac{s_A^2}{n_A} + \frac{s_B^2}{n_B}}}$$

Use Welch's t-test: Unequal variances OK

Confidence Intervals

95% CI for difference:

$$(\bar{x}_B - \bar{x}_A) \pm 1.96 \cdot SE$$

Interpretation:

- If CI excludes 0

$\rightarrow$ Significant

- CI shows practical range
- More informative than p-value alone

Example:

- Difference: 1.2 percentage points
- 95% CI: [0.8, 1.6]
- Excludes 0

$\rightarrow$ Significant

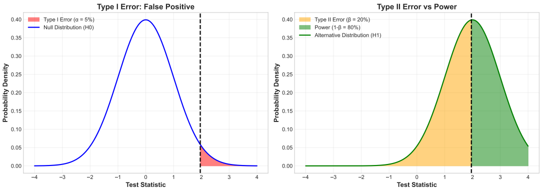
- Likely between 0.8-1.6pp lift

Effect Size

Cohen's d:

$$d = \frac{\bar{x}_B - \bar{x}_A}{s_{pooled}}$$

Statistical Significance: Interpreting P-Values



What is a P-Value?

Definition: Probability of observing data this extreme if null hypothesis were true.

NOT:

- Probability null is true
- Probability you're wrong
- Size of effect

Example:

- $p = 0.03$
- If no real difference exists, 3% chance of seeing this result by luck
- Since $3\% < 5\%$, reject null

Type I vs Type II Errors

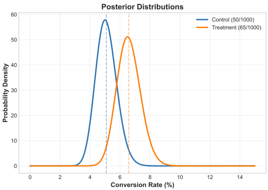
Type I (False Positive):

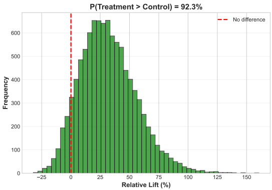
- Declare winner when none exists
- Rate: α (typically 5%)
- "Ship a loser"

Type II (False Negative):

- Miss real improvement

Bayesian A/B Testing: Probability of Being Best





Bayesian Approach

Instead of p-values:

- Calculate $P(B > A)$
- Direct probability interpretation
- Incorporate prior knowledge
- No arbitrary 0.05 threshold

Example:

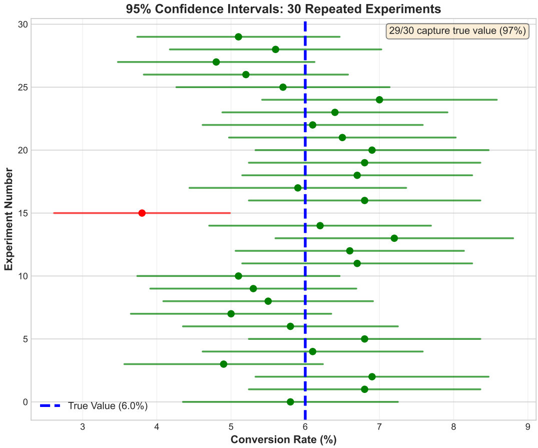
- $P(B > A) = 0.97$
- 97% sure treatment is better
- Can ship with 95% confidence

Advantages

- Intuitive interpretation
- Earlier stopping (faster decisions)
- Handles small samples better
- Can update continuously
- No "peeking problem"

When to Use

Understanding Confidence Intervals Through Repetition



What 95% CI Means

Definition: If we repeated the experiment 100 times, 95 of the intervals would contain the true value.

Key Insights:

- Each experiment produces a different interval
- True value is fixed (but unknown)
- Some intervals miss (by design!)
- Wider intervals = more uncertainty

Common Misconceptions

Wrong: "95% chance true value is in this interval"

- True value either is or isn't in it
- It's about the procedure, not this interval

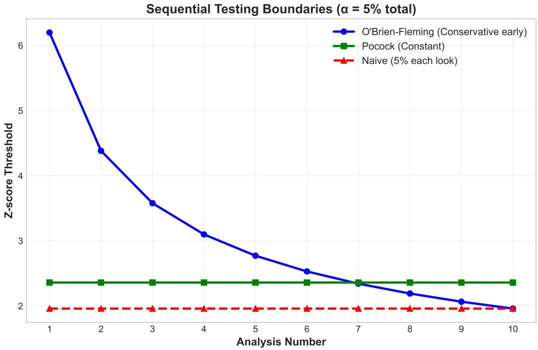
Right: "95% of intervals constructed this way contain the true value"

Practical Use

- If CI includes 0: Not significant
- If CI excludes 0: Significant
- Width shows precision of estimate

Repeated sampling visualization clarifies confidence interval interpretation - seeing multiple intervals reveals the procedure's long-run behavior

Sequential Testing: Early Stopping Without Peeking



The Peeking Problem

Bad practice:

- Check results daily
- Stop when p $<$ 0.05
- False positive rate $>$ 50%!

Why bad:

- Multiple testing inflates α
- Random fluctuations look significant
- Results not reproducible

Sequential Testing Solution

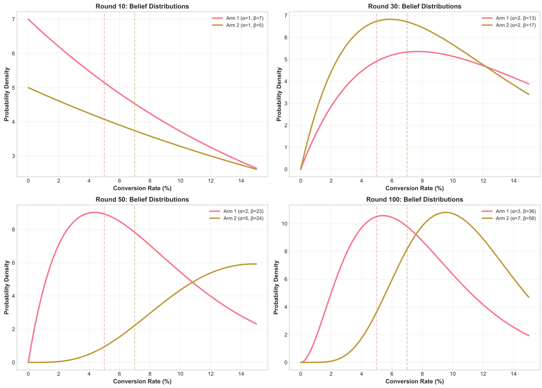
Method: Alpha spending functions

- Plan interim analyses (e.g., 25%, 50%, 75%, 100%)
- Adjust α at each look
- Control overall Type I error

O'Brien-Fleming:

- Conservative early, liberal late
- First look: $\alpha = 0.0005$
- Last look: $\alpha = 0.0455$

Multi-Armed Bandits: Exploration vs Exploitation



The Bandit Problem

Scenario: Slot machines (arms) with unknown payouts.
Goal: Maximize total reward.

Trade-off:

- **Explore:** Try arms to learn
- **Exploit:** Use best-known arm

Thompson Sampling

Algorithm:

1. Maintain Beta(α, β) for each arm
2. Sample from each distribution
3. Pick arm with highest sample
4. Update winner's distribution

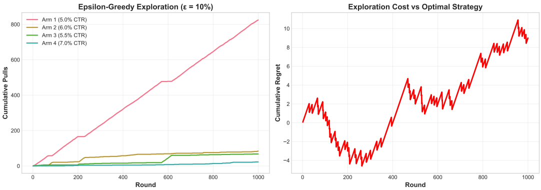
Why it works:

- Automatically balances exploration/exploitation
- Converges to best arm
- Minimizes regret

A/B Test vs Bandit

A/B Test:

Bandit Performance: Exploration vs Regret



How Bandits Learn

Left Panel: Cumulative Pulls

- Initially: Explore all arms
- Middle: Identify good arms
- End: Exploit best arm
- Bad arms stop getting traffic

Right Panel: Regret

Regret: Total reward lost by not always picking best arm

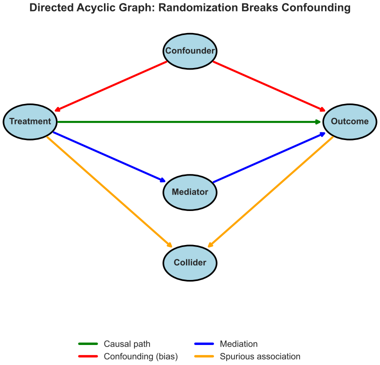
Key Insight:

- Regret grows logarithmically
- Much slower than linear
- Thompson Sampling is near-optimal
- A/B test has linear regret

When to Use Bandits

- High traffic (fast learning)
- Multiple variants (> 3)
- Cost of regret is high
- Can tolerate adaptive allocation

Causal Inference: Correlation is Not Causation



The Fundamental Problem

Correlation: X and Y move together
Causation: X causes Y

Why different:

- Confounders (Z affects both X and Y)
- Reverse causation (Y causes X)
- Spurious correlation (coincidence)

Example: Ice Cream & Drowning

- Correlation: High
- Causation: None
- Confounder: Hot weather
- Weather causes both ice cream sales and swimming (drowning risk)

Randomization Solves This

How:

- Random assignment breaks confounding
- Only difference is treatment
- Can infer causation

Multi-Objective Optimization: Trade-Offs

The Real-World Problem

Single metric is rare:

- CTR vs revenue
- Short-term vs long-term
- Engagement vs satisfaction
- Speed vs accuracy

Example: Recommendation algorithm

- Control: Higher revenue, lower engagement
- Treatment: Lower revenue, higher engagement
- Which is better?

Solution 1: Primary + Guardrails

Method:

- Primary: Metric you optimize (revenue)
- Guardrails: Metrics that must not degrade
- Example: Maximize revenue, but engagement must not drop $> 2\%$

Decision rule:

- Ship if primary improves AND guardrails met

Solution 2: Weighted Utility

Method: Combine metrics into single score

$$U = w_1 \cdot revenue + w_2 \cdot engagement$$

Example weights:

- Revenue: 0.7 (70% weight)
- Engagement: 0.3 (30% weight)

Challenge: Choosing weights

Solution 3: Pareto Frontier

Method: Find non-dominated solutions

- Solution A dominates B if better on all metrics
- Pareto frontier: Set of non-dominated solutions
- Business decides among frontier

When to use:

- Cannot agree on weights
- Want to see trade-off space
- Explore multiple strategies

Technique Comparison: When to Use Each

Classical A/B Test

When:

- High traffic
- Need causal rigor
- Low risk
- Regulatory requirements
- Can wait for full sample

Pros:

- Rigorous
- Well understood
- Easy to interpret
- Reproducible

Cons:

- Slow (fixed duration)
- Higher regret
- Can't peek

Example:
Major algorithm change, need 95%

Bayesian A/B Test

When:

- Need faster decisions
- Low traffic
- Can update continuously
- Prior knowledge exists
- Stakeholders prefer probabilities

Pros:

- Intuitive ($P(B > A)$)
- Earlier stopping
- Handles small samples
- No peeking problem

Cons:

- Prior selection subjective
- Less familiar to stakeholders
- Requires Bayesian tooling

Example:
Startup with 1K daily users, need quick wins

Multi-Armed Bandit

When:

- Minimize regret
- Continuous optimization
- Many variants (A/B/C/D/E)
- Cost of losing high
- Real-time personalization

Pros:

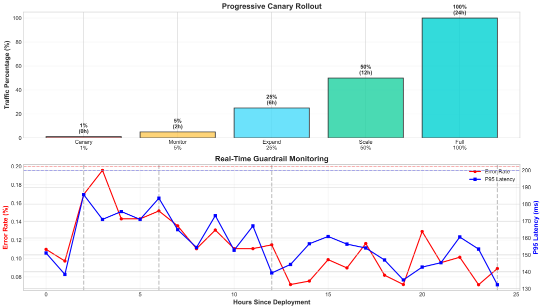
- Lowest regret
- Adaptive allocation
- Scales to many arms
- Always optimizing

Cons:

- Less statistical rigor
- Harder to interpret
- Complex implementation

Example:
Ad creative testing (50 variants), email

Production Experiment Infrastructure



Core Components

1. **Traffic Splitting**
   - Assign users to control/treatment
   - Consistent hashing (same user, same group)
   - 50/50, 90/10, or custom splits
2. **Feature Flags**
   - Toggle experiments on/off
   - Gradual rollouts (1% $\rightarrow$ 5% $\rightarrow$ 25% $\rightarrow$ 100%)
   - Instant kill switch
3. **Experiment Tracking**
   - Log user assignment
   - Track metrics per group
   - Store for analysis
4. **Analysis Pipeline**
   - Automated statistical tests
   - Real-time dashboards
   - Alert on guardrail violations

Experiment infrastructure requires orchestration components - traffic splitting, feature flags, tracking, and analysis form the core system

Classical A/B Test Implementation:

- Import required libraries: `scipy.stats`, `statsmodels`
- Define sample data: control and treatment groups
- Calculate proportions for each group
- Perform Z-test for proportions using `proportions_ztest`
- Compare p-value to significance threshold (0.05)
- Calculate and display key metrics:
    - Control rate, treatment rate
    - Lift percentage
    - Z-statistic and p-value
    - Significance decision

Output Example

Sample Output:

- Control: 0.050 (5.0% conversion)
- Treatment: 0.062 (6.2% conversion)
- Lift: 24.0% improvement
- Z-statistic: 4.72
- P-value: 0.0001
- Result: Significant improvement!

Confidence Interval

Confidence Interval Calculation:

- Import `confint_proportions_2indep` from statsmodels
- Calculate 95% confidence interval using Wald method
- Parameters: treatment successes/total, control successes/total
- Output: lower and upper bounds of difference

Interpretation

If CI excludes 0, treatment significantly better than control

Statistical libraries provide essential testing infrastructure - robust implementations enable reliable hypothesis evaluation

Python: Bayesian A/B Testing with PyMC3

Bayesian A/B Test with PyMC3:

- Import PyMC3 and numpy libraries
- Define observed data: clicks and trials for groups A and B
- Set up Bayesian model:
  - Beta priors for conversion rates (non-informative)
  - Binomial likelihoods for observed data
  - Deterministic variable for difference (delta)
- Sample from posterior distribution (2000 samples)
- Calculate probability that B outperforms A
- Direct probability statements without p-value confusion

Output

Bayesian Results:

- $P(B > A) = 0.997$ (99.7% confidence)
- Posterior mean delta: 0.012
- 95% Credible Interval: [0.008, 0.016]

Interpretation

- 99.7% probability B is better
- Expected lift: 1.2 pp
- 95% sure lift between 0.8-1.6pp
- High confidence to ship

Advantages

- Direct probability statements
- No p-value confusion
- Can stop early if $P(B > A) > 0.95$
- Incorporates prior knowledge

Probabilistic inference enables faster experimental decisions - direct probability statements reduce interpretation overhead

Python: Thompson Sampling Bandit

Thompson Sampling Bandit Implementation:

- Class initialization with number of arms
- Maintain success/failure counts per arm (Beta parameters)
- Selection algorithm:
  - Sample from Beta distribution for each arm
  - Select arm with highest sample value
- Update mechanism:
  - Increment success count for reward $= 1$
  - Increment failure count for reward $= 0$
- Main loop: select arm, observe reward, update beliefs
- Converges to optimal arm while balancing exploration

How It Works

1. **Initialize:**
   - Each arm: Beta(1, 1) = uniform prior
2. **Select arm:**
   - Sample from each Beta distribution
   - Pick arm with highest sample
3. **Observe reward:**
   - If success: $\alpha + 1$
   - If failure: $\beta + 1$
4. **Repeat**

Convergence

- Best arm gets pulled more often
- Exploration decreases over time
- Minimizes cumulative regret

Use case: Ad creative testing (50 variants, continuous optimization)

Probabilistic selection algorithms balance simplicity and effectiveness - sampling from posterior distributions automates exploration-exploitation trade-offs

Python: Stratified Randomization with sklearn

Stratified Randomization Implementation:

- Import `StratifiedShuffleSplit` from sklearn
- Create user dataframe with stratification variables
- Set up stratified split:
  - 50/50 split between control and treatment
  - Stratify by device type (iOS/Android)
  - Use fixed random state for reproducibility
- Apply split to create balanced groups
- Verify stratification worked:
  - Check device distribution in both groups
  - Should be identical proportions
- Can extend to multiple stratification variables

Output

Verification Results:

- **Control device distribution:**
  - iOS: 0.50 (50%)
  - Android: 0.50 (50%)
- **Treatment device distribution:**
  - iOS: 0.50 (50%)
  - Android: 0.50 (50%)

Why This Matters

Without stratification:

- Control might be 55% iOS
- Treatment might be 45% iOS
- Device becomes confounder
- Results biased

With stratification:

- Both groups exactly 50% iOS
- Device balanced
- Removes confounding
- Higher statistical power

Python: Sequential Testing with Early Stopping

Sequential Testing with O'Brien-Fleming:

- Import numpy and scipy.stats.norm
- Define O'Brien-Fleming boundary function:
  - Input: current analysis number (k), total analyses (K)
  - Calculate alpha spending based on information fraction
  - Conservative early, liberal late
- Plan interim analyses (e.g., 25%, 50%, 75%, 100%)
- For each analysis:
  - Calculate current p-value
  - Compare to adjusted alpha threshold
  - Stop if significant, continue otherwise
- Maintains overall Type I error rate at 0.05
- Enables early stopping without "peeking" penalty

Alpha Spending

Alpha Spending Schedule:

- Look 1 (25%): alpha = 0.0005
- Look 2 (50%): alpha = 0.0139
- Look 3 (75%): alpha = 0.0303
- Look 4 (100%): alpha = 0.0455

Interpretation

- Very conservative early (0.0005)
- More liberal late (0.0455)
- Total α still 0.05
- Can stop early if clear winner

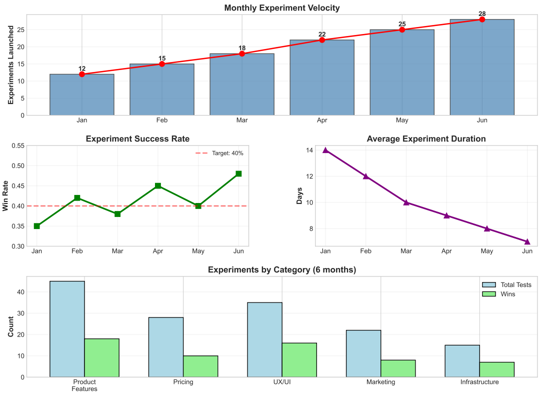
Benefits

- Valid p-values at each look
- Can stop for futility (clear loser)
- Faster decisions on average
- Maintains Type I error control

Cost: Slightly larger sample needed vs fixed design

Alpha spending enables valid interim analyses - pre-planned looks distribute error rate across sequential observations

Real-Time Experiment Monitoring



Key Metrics to Monitor

Primary Metric:

- Current estimate
- Confidence interval
- P-value or P(B $>$ A)
- Sample size achieved

Guardrail Metrics:

- Revenue (must not drop $>$ 2%)
- Latency (p95 $<$ 200ms)
- Error rate ($<$ 0.1%)
- Automatic alerts if violated

Diagnostic Metrics:

- Sample ratio mismatch
- Traffic allocation balance
- Novelty effects
- Time-of-day patterns

Automated Actions

- Stop if guardrail violated

Common A/B Testing Pitfalls

Peeking Problem

What: Check results daily, stop when $p < 0.05$

Why bad:

- Inflates false positive rate
- Random fluctuations look significant
- Not reproducible

Fix:

- Wait for planned sample
- Use sequential testing
- Or Bayesian methods

Multiple Testing

What: Test 20 variants, report winner

Why bad:

- 1 false positive expected
- Winner likely spurious

Fix:

- Bonferroni correction

Simpson's Paradox

What: Treatment wins overall, loses in every segment

Example:

- iOS: Control 10%, Treatment 9%
- Android: Control 8%, Treatment 7%
- Overall: Treatment wins (imbalance in device mix)

Fix:

- Stratified randomization
- Analyze by segment
- Check for confounders

Novelty Effects

What: Users react to change itself, not treatment

Signs:

- Week 1: Treatment wins
- Week 2: Effect shrinks
- Week 3: No difference

Sample Ratio Mismatch

What: 50/50 split becomes 52/48

Causes:

- Bot traffic
- Implementation bugs
- Sampling bias

Why bad:

- Groups not comparable
- Results invalid

Fix:

- Monitor ratio daily
- Investigate deviations
- Filter bots

Ignoring Variance

What: Focus only on means, ignore spread

Why bad:

- High variance = unreliable
- May hurt some users badly

Production Deployment Patterns

Canary Release

Process:

1. Deploy to 1% users
2. Monitor for 24 hours
3. If stable, 5%
4. Then 25%, 50%, 100%

When:

- High-risk changes
- New algorithms
- Large refactors

Benefits:

- Early error detection
- Minimal blast radius
- Easy rollback

Guardrails:

- Error rate $< 0.5\%$
- Latency $<$ baseline $+ 20\%$
- Revenue $>$ baseline - 5%

Blue-Green Deploy

Process:

1. Green: Current (100%)
2. Blue: New (0%)
3. Gradually shift traffic
4. Blue becomes 100%

When:

- Infrastructure changes
- Database migrations
- Zero-downtime deploys

Benefits:

- Instant rollback (flip traffic)
- Both versions running
- Compare metrics live

Challenges:

- 2*imes* infrastructure cost
- Data consistency
- State management

Shadow Mode

Process:

1. Send traffic to both models
2. Serve old model to users
3. Log new model predictions
4. Compare offline

When:

- Validating new model
- Zero risk testing
- Performance benchmarking

Benefits:

- No user impact
- Real traffic testing
- Can run indefinitely

Use case:

- Test before A/B test
- Validate offline metrics
- Check for bugs

Implementation Summary & Best Practices

Implementation Checklist

Before Experiment:

- Hypothesis pre-registered
- Sample size calculated
- Randomization strategy chosen
- Guardrails defined
- Monitoring dashboard ready
- Kill switch tested

During Experiment:

- Monitor metrics daily
- Check sample ratio balance
- Watch for guardrail violations
- No peeking at p-values (unless sequential)
- Document any issues

After Experiment:

- Run statistical tests
- Calculate confidence intervals
- Check subgroup analyses

Python Tools Ecosystem

Statistical Testing:

- scipy: t-tests, z-tests
- statsmodels: Advanced tests, power analysis
- PyMC3: Bayesian methods

Experimentation Platforms:

- GrowthBook: Open-source, feature flags
- Optimizely: Enterprise A/B testing
- LaunchDarkly: Feature management
- Firebase: Mobile experiments

Visualization:

- matplotlib/seaborn: Static plots
- Plotly: Interactive dashboards
- Streamlit: Quick apps

Key Takeaways

- Automate as much as possible
- Monitor continuously
- Plan for early stopping

Building Experimentation Culture: Leadership Principles

Jeff Bezos on Experimentation

"If you double the number of experiments you do per year, you're going to double your inventiveness."

Amazon's Approach:

- Experiments are default, not exception
- Ship fast, iterate faster
- Celebrate "failed" experiments (learning)
- Data beats opinions
- Everyone empowered to test

Google's Philosophy

"We're wrong a lot. Most experiments fail. That's OK—we learn fast and move on."

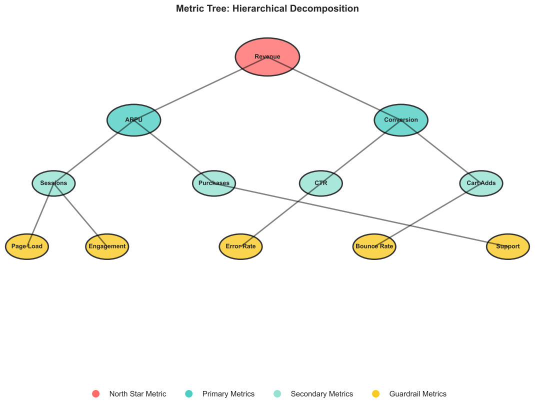
Key principles:

- 10-20% win rate is normal
- Compound small wins (1% lift $\times$ 100 experiments)
- Avoid HiPPO (Highest Paid Person's Opinion)
- Test everything, assume nothing

Characteristics of Strong Culture

1. **Velocity:**
   - 50-100+ experiments per year
   - 1-2 week experiment cycle
   - Automated pipelines
2. **Psychological Safety:**
   - OK to propose "crazy" ideas
   - No punishment for failed experiments
   - Learn from mistakes openly
3. **Data Literacy:**
   - Understand p-values, CI, effect sizes
   - Know when to trust results
   - Question unexpected findings
4. **Infrastructure:**
   - Feature flags in production
   - Real-time dashboards
   - 1-click deploy and rollback

Metric Selection: North Star & Guardrails



North Star Metric

Definition: Single metric that captures long-term value creation

Examples:

- Spotify: Weekly active users
- Netflix: Watch time per user
- Airbnb: Nights booked
- Amazon: Purchases per customer

Criteria:

- Aligns with business model
- Measurable in A/B tests
- Leading indicator (not lagging)
- Actionable by teams

Guardrail Metrics

Purpose: Prevent optimization tunnel vision

Examples:

- Revenue: Must not drop $> 2\%$
- Latency: p95 $<$ 300ms

Communicating Results to Stakeholders

For Executives

What they care about:

- Bottom line impact
- Risk level
- Confidence level
- Timeline

Example:
"Treatment increased revenue by 8% (95% CI: 5%-11%). Expected annual impact: $2.4M. Recommend full rollout over 2 weeks. Risk: Low (guardrails passed)."

For Product Managers

What they care about:

- User experience impact
- Segment differences
- Feature interactions
- Next iterations

Example:
"Treatment improved CTR 15% for new users, 5% for power users. iOS stronger than

For Engineers

What they care about:

- Implementation details
- Performance metrics
- Technical challenges
- Edge cases

Example:
"Model latency p95: 180ms (vs 150ms baseline). Sample ratio 50.2/49.8 (acceptable). Bug in iOS < 13 handling fixed day 3. Rerun analysis confirmed results."

For Data Scientists

What they care about:

- Statistical rigor
- Effect sizes
- Confidence intervals
- Methodology

Example:
"$Z = 4.2$, $p < 0.001$, Cohen's $d = 0.28$

Universal Principles

Always include:

1. Hypothesis tested
2. Sample size achieved
3. Primary metric result + CI
4. Guardrail metric status
5. Recommendation (ship/iterate/kill)
6. Rationale for recommendation

Visualization Tips

- Show distributions, not just means
- Include confidence intervals
- Highlight guardrail bounds
- Use color coding (green/yellow/red)
- Keep it simple (no jargon)

Red Flags to Call Out

- Sample ratio mismatch
- Novelty effects
- Simpson's paradox

Explaining Statistics to Non-Technical Stakeholders

P-Value Translation

Instead of:
"P-value is 0.03, so we reject the null hypothesis at $\alpha = 0.05$."

Say:
"If there were truly no difference, we'd see a result this extreme only 3% of the time by random chance. Since that's unlikely, we're confident treatment is better."

Confidence Interval Translation

Instead of:
"95% CI: 1.2%-3.8%"

Say:
"We're 95% confident the true improvement is between 1.2% and 3.8%. Best estimate: 2.5%."

Statistical Power Translation

Instead of:
"80% power to detect 0.5pp effect"

Say:
"If there's a real 0.5 percentage point improvement, we'll detect it 80% of the time. We might miss smaller improvements."

Common Misunderstandings

Wrong: "P = 0.03 means 97% chance treatment is better"

- P-value is NOT probability hypothesis is true
- Use Bayesian methods for that

Wrong: "Significant = Important"

- Statistical $\neq$ Practical significance
- 0.01% lift might be significant but meaningless

Wrong: "Not significant = No effect"

- Absence of evidence $\neq$ Evidence of absence
- Might just need larger sample

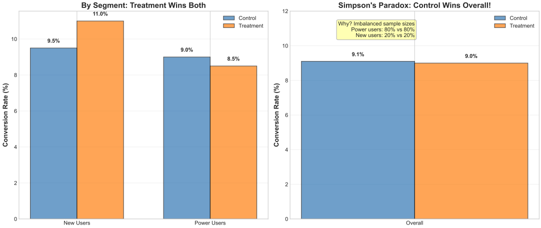
Analogies That Work

Confidence Interval:
"Like a weather forecast: 70-80°F tomorrow means we're sure it won't be 50° or 100°, but exact temp uncertain."

Type I/II Errors:
"Fire alarm: False alarm (Type I) = evacuate unnecessarily. Missed fire (Type II) = danger goes undetected."

Statistical literacy compounds across iterations - understanding accumulates faster than individual technique mastery

Simpson's Paradox: When Aggregation Misleads



The Paradox

What happens:

- Treatment wins in segment A
- Treatment wins in segment B
- Control wins overall!

Why: Imbalanced sample sizes across segments distort the aggregate result.

Real Example

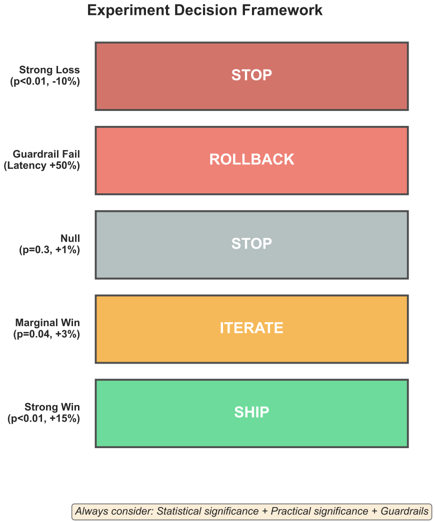
UC Berkeley Admission (1973):

- Overall: Men 44%, Women 35% (bias?)
- But in most departments: Women had higher admit rate
- Explanation: Women applied to harder departments

How to Detect

- Always analyze by key segments
- Check if segment sizes balanced
- Use stratified randomization
- Report both aggregate and segment results

Decision Frameworks: Ship, Iterate, or Kill?



Ship Criteria

All must be true:

- Statistically significant ($p < 0.05$ or $P(B > A) > 0.95$)
- Practically significant (effect size meaningful)
- All guardrails passed
- Benefit $>$ cost (ROI positive)
- Implementation ready

Example:
15% CTR lift, $p = 0.001$, \$500K annual revenue, latency OK, error rate OK $\rightarrow$ SHIP

Iterate Criteria

When to iterate:

- Directionally positive but not significant
- Significant but guardrail violated
- Strong in 1 segment, weak in others
- Good idea, poor execution

Example:
3% CTR lift, $p = 0.12$ (not sig), but iOS showed 8% lift ($p = 0.01$) $\rightarrow$ ITERATE for iOS only

Decision criteria reduce deliberation overhead - pre-specified thresholds eliminate recursive argumentation

Measuring Long-Term Impact: Beyond the Experiment

The Long-Term Challenge

Problem:

- Experiments run 1-2 weeks
- But true impact unfolds over months
- Short-term vs long-term trade-offs

Example:

- Treatment boosts Week 1 engagement +20%
- Week 4: Back to baseline (novelty wore off)
- Month 6: -5% (user fatigue)

Solution 1: Holdout Groups

Method:

- Keep 10% of users in control forever
- Compare treatment vs holdout over 6-12 months
- Measure long-term metrics

Benefits:

- Detect delayed effects
- Catch cumulative degradation
- Measure true business impact

Solution 2: Cohort Analysis

Method:

- Track users by week of experiment entry
- Compare Week 1, 2, 3 cohorts over time
- Look for retention/engagement patterns

Example:

Week 1 cohort: 30-day retention 60%
Week 4 cohort: 30-day retention 55% → Novelty effect

Solution 3: Synthetic Controls

Method:

- Find similar users not in experiment
- Match on behavior pre-experiment
- Compare long-term trajectories

Use case:

- When holdout not feasible
- Geographic experiments
- Policy changes

Key Metrics to Track

Ethical Experimentation: Do No Harm

Core Principles

1. Informed Consent

- Users should know they're in experiments
- Privacy policy disclosure
- Opt-out mechanisms

2. Minimize Harm

- Don't test things that could hurt users
- Set guardrails (revenue, UX, safety)
- Monitor closely for negative effects
- Have kill switch ready

3. Fairness

- Don't systematically disadvantage groups
- Check for disparate impact
- Ensure equal treatment opportunity

4. Transparency

- Document methodology
- Share learnings internally
- Be honest about failures

What NOT to Test

Unethical Experiments:

- Intentionally degrade user experience
- Manipulate emotions without consent
- Test discriminatory policies
- Hide critical information
- Exploit vulnerable populations

Famous Missteps:

- Facebook emotion study (2014): Manipulated newsfeeds to study emotional contagion without consent
- OkCupid compatibility test: Lied about match scores to see behavioral effects

Best Practices

- Ethics review for sensitive experiments
- Independent oversight for high-risk tests
- User research panel for feedback
- Regular audits of experiment portfolio
- Clear escalation paths for concerns

Experiment Velocity: The Compounding Effect

Why Velocity Matters

Scenario 1: Slow Team

- 10 experiments per year
- 30% win rate
- 3 wins per year
- Average lift per win: 5%
- Annual improvement: 15% (additive)

Scenario 2: Fast Team

- 100 experiments per year
- 30% win rate (same)
- 30 wins per year
- Average lift per win: 1% (smaller but more)
- Annual improvement: 35% (compounding)

The Math

$(1.01)^{30} = 1.35$ (35% improvement)
Many small wins compound faster than few large wins

Google's approach: Would rather ship 100 experiments with 1% lifts than 10 with 10% lifts

Bottlenecks to Address

1. **Slow Implementation**
   - Solution: Feature flags, modular code
   - Goal: Idea $\rightarrow$ live experiment in 2 days
2. **Long Experiment Duration**
   - Solution: Higher traffic, sequential testing
   - Goal: Results in 1-2 weeks max
3. **Slow Analysis**
   - Solution: Automated pipelines, dashboards
   - Goal: Results available real-time
4. **Decision Paralysis**
   - Solution: Clear criteria, empowered teams
   - Goal: Decision within 24 hours of results

Target Metrics

- Experiments per engineer per quarter: 5-10
- Average experiment duration: 1-2 weeks
- Time to decision: 1-3 days

Tools & Infrastructure for Experimentation at Scale

Open Source

GrowthBook

- Feature flags + A/B testing
- Bayesian statistics
- SQL-based metrics
- Self-hosted or cloud

Unleash

- Feature toggle management
- Gradual rollouts
- Real-time updates
- Lightweight

Statsig (Free tier)

- Feature gates
- Dynamic configs
- Autotune
- Built-in metrics

DIY Stack

- Feature flags: Flagsmith, ConfigCat

Enterprise

Optimizely

- Market leader
- Full-stack experimentation
- Visual editor
- Expensive ($50K+/year)

LaunchDarkly

- Feature management focus
- Targeting rules
- Fast flag updates
- Popular with DevOps

VWO

- Web optimization
- Heatmaps, session replay
- A/B + multivariate
- Marketing-friendly

Firebase A/B Testing

- Mobile-first

Big Tech Internal

Google's Dapper

- 1000s of concurrent experiments
- Automated metrics computation
- Novelty detection
- Not public

Facebook's Gatekeeper

- Feature gating
- Dynamic allocation
- Layered experiments
- Internal only

Build vs Buy

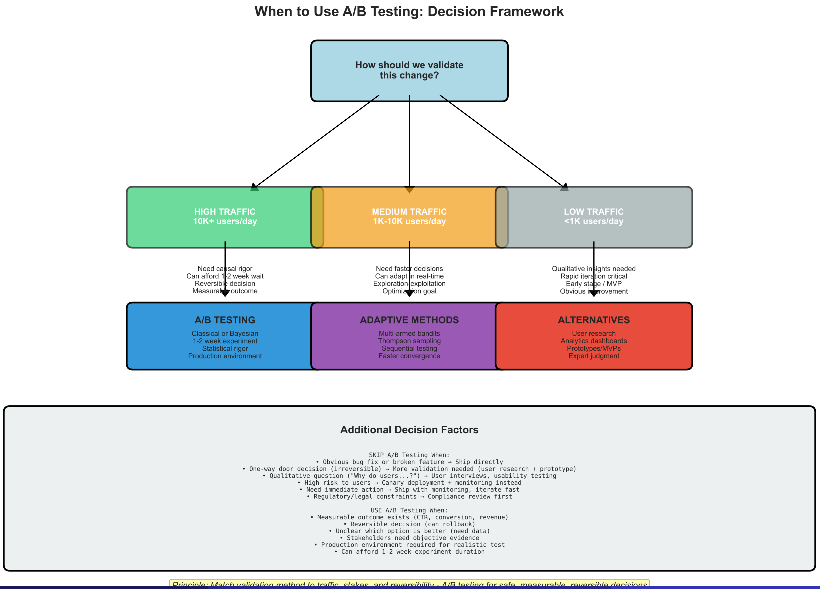
Buy (SaaS) if:

- Small team (< 10 engineers)
- Need to move fast
- Standard use cases

Build (Custom) if:

- Large scale (1M+ users)

When to Use A/B Testing: Judgment Criteria



Design & Culture Summary

Key Principles

1. Culture

- Experiments are default, not exception
- Psychological safety for "failed" tests
- Data beats opinions (no HiPPO)
- Celebrate learning, not just wins

2. Metrics

- North Star: Long-term value proxy
- Guardrails: Prevent negative side effects
- Leading indicators, not lagging

3. Communication

- Tailor to audience (exec/PM/eng/DS)
- Always include CI + effect size
- Plain English, no jargon
- Visual ¿ numbers

4. Decision-Making

- Clear ship/iterate/kill criteria
- Fast decisions (within 24 hours)

Strategic Insights

5. Long-Term Thinking

- Holdout groups for 6-12 month measurement
- Cohort analysis for retention
- Don't optimize away future value

6. Ethics

- Informed consent
- Minimize harm
- Fairness across groups
- Transparency

7. Velocity

- 50-100 experiments per year (target)
- Compound small wins
- Reduce bottlenecks
- Automate everything possible

8. Infrastructure

- Feature flags in production
- Real-time dashboards

Workshop: E-Commerce Recommendation Engine Comparison

Your Challenge

Design and analyze an A/B test comparing 3 recommendation algorithms for an e-commerce site with 100,000 daily users.

Why This Matters:

- Real-world ML deployment decision
- Multi-model comparison
- Statistical rigor + business alignment
- Portfolio project for interviews

Success Criteria:

- Correct sample size calculation
- Proper statistical tests applied
- Guardrail metrics checked
- Clear recommendation with rationale
- Rollout plan with risk mitigation

What You'll Do

1. Design experiment (hypothesis, metrics, sample size)
2. Conduct power analysis (duration calculation)
3. Run simulation (3 models, 30K users)
4. Perform statistical analysis (t-tests, Bayesian)
5. Check guardrails (revenue, latency)
6. Make deployment decision (ship/iterate/kill)
7. Create rollout plan (1% $\rightarrow$ 100%)

Time: 60 minutes
Deliverable: Jupyter notebook
Format: Individual or pairs
Tools: Python (scipy, PyMC3)

Validation establishes baseline quality while iteration drives improvement - measurement cycles enable optimization

Business Context & Baseline Performance

Current State

E-Commerce Site:

- 100,000 daily active users
- 1M product catalog
- Average order value: $50
- Revenue: $2M per day

Existing Recommendation System:

- Algorithm: Popularity-based
- CTR (click-through rate): 5.0%
- Conversion rate: 2.0%
- Revenue per user: $20
- Latency: p95 = 150ms

Business Goal:

- Increase engagement (CTR)
- Maintain or improve conversion
- Don't degrade revenue
- Keep latency < 200ms

Candidate Algorithms

Model A: Collaborative Filtering (CF)

- Offline CTR estimate: 6.2%
- Conversion: 2.1% (expected)
- Latency: 180ms
- Complexity: Medium

Model B: Content-Based (CB)

- Offline CTR estimate: 5.8%
- Conversion: 2.0% (expected)
- Latency: 160ms
- Complexity: Low

Model C: Hybrid (CF + CB)

- Offline CTR estimate: 6.5%
- Conversion: 2.2% (expected)
- Latency: 190ms
- Complexity: High

Key Question

Which model should we deploy? Or should we iterate further?

Task 1: Experiment Design (10 minutes)

Hypothesis Formulation

Task: Write 3 hypotheses (one per model)

Example (Model A - CF):
"Switching from popularity-based to collaborative filtering will increase CTR by at least 1 percentage point (from 5% to 6%) over a 2-week test with 100,000 users, without decreasing revenue per user by more than 2%."

Primary Metric

Choose one:

- Click-through rate (CTR)
- Conversion rate
- Revenue per user
- Engagement time

Recommendation: CTR

- Most sensitive (changes fastest)
- Leading indicator for conversion/revenue
- Easier to detect statistically

Guardrail Metrics

Task: Define guardrails

Recommended:

- Revenue per user: Must not drop $> 2\%$
- Latency p95: Must stay < 200ms
- Error rate: Must stay $< 0.1\%$
- Conversion rate: Must not drop > 0.2pp

Experimental Design

Traffic Allocation:

- Control (Popularity): 25%
- Treatment A (CF): 25%
- Treatment B (CB): 25%
- Treatment C (Hybrid): 25%

Randomization:

- User-level (consistent experience)
- Stratified by device (iOS/Android)
- Hash-based assignment

Duration: TBD (power analysis next)

Task 2: Power Analysis (10 minutes)

Sample Size Calculation

Parameters:

- Baseline CTR: 5% (p_1)
- Target CTR: 6% (p_2)
- MDE (Minimum Detectable Effect): 1pp
- Significance level (α): 0.05
- Statistical power (1 - β): 0.80

Formula (proportions test):

$$n = \frac{(z_{\alpha/2} + z_\beta)^2 \cdot [p_1(1 - p_1) + p_2(1 - p_2)]}{(p_2 - p_1)^2}$$

Calculation:

- $z_{0.025} = 1.96$, $z_{0.20} = 0.84$
- $n \approx$ 9,800 per group
- 4 groups $\rightarrow$ 39,200 total users

Duration:

- Daily users: 100,000
- Required: 39,200

Python Implementation

Code structure (see notebook):

- Import statsmodels.stats.power
- Import numpy for calculations
- Define baseline CTR: 0.05
- Define target CTR: 0.06
- Set alpha: 0.05, power: 0.80
- Calculate effect size (Cohen's h)
- Use zt_ind_solve_power function
- Compute sample size per group

Expected Output

Sample size calculations:

- Sample size per group: 9,800
- Total sample needed: 39,200
- Days needed: 0.39
- Recommendation: Run for 14 days to capture weekly patterns

Key insight: Always run longer than statistical minimum to avoid novelty effects and capture weekly seasonality

Task 3: Simulation (15 minutes)

Generate Synthetic Data

Task: Simulate user interactions

Setup:

- 30,000 users (more than minimum for confidence)
- 7,500 per group
- Bernoulli trials (click = 1, no click = 0)

True CTRs (ground truth):

- Control: 5.0%
- Treatment A (CF): 6.2%
- Treatment B (CB): 5.8%
- Treatment C (Hybrid): 6.5%

Python Simulation Steps

Implementation (see notebook):

- Import numpy, set random seed
- Set n_per_group = 7500
- Generate control clicks: binomial(1, 0.050, n)
- Generate treatment_a: binomial(1, 0.062, n)
- Generate treatment_b: binomial(1, 0.058, n)

Expected Output

Observed CTRs:

- Control: 0.051
- Treatment A: 0.061
- Treatment B: 0.058
- Treatment C: 0.066

Observations:

- Slight variation from true CTR (sampling noise)
- Treatment C highest (6.6%)
- Treatment A second (6.1%)
- Treatment B marginal (5.8%)

Add Guardrail Metrics

Simulate additional metrics:

- Revenue per user: normal distribution
- Control mean: $20, std: $5
- Treatment A: $20.50, std: $5
- Treatment B: $20, std: $5
- Treatment C: $21, std: $5

Task 4: Statistical Analysis (15 minutes)

Classical Z-Test

Task: Compare each treatment vs control

Implementation steps:

- Import proportions_ztest from statsmodels
- For each treatment (A, B, C):
  - Create count array: [treatment.sum(), control.sum()]
  - Create nobs array: [len(treatment), len(control)]
  - Run proportions_ztest(count, nobs)
  - Extract z-statistic and p-value
  - Compare p-value to alpha = 0.05
- Print results with interpretation

Expected Results:

- A vs Control: $p < 0.001$ (significant)
- B vs Control: $p \approx 0.08$ (not significant)
- C vs Control: $p < 0.001$ (significant)

Bayesian Analysis

Task: Calculate P(Treatment > Control)

PyMC3 implementation:

- Import pymc3, create model context
- Define Beta(1,1) priors for p_control
- Define Beta(1,1) priors for p_treatment_c
- Add Binomial likelihood for control
- Add Binomial likelihood for treatment_c
- Sample posterior with 2000 draws
- Calculate probability: treatment > control
- Compute mean of boolean comparison

Expected Result:

- $P(C > \text{Control}) = 0.998$
- Interpretation: 99.8% confidence C is better

Convergence

Both frequentist (z-test) and Bayesian methods agree: Treatment C is the clear winner

Methodological convergence strengthens conclusions - independent analytical approaches yielding consistent results increase confidence

Task 5: Guardrail Check & Decision (10 minutes)

Guardrail Metrics Check

Task: Verify all guardrails passed

Treatment C (Hybrid) - Winner Candidate:

Check 1: CTR Lift

- ctr_control = control.mean()
- ctr_c = treatment_c.mean()
- lift = (ctr_c / ctr_control - 1) * 100
- Result: 29% lift

Check 2: Revenue

- rev_control = revenue_control.mean()
- rev_c = revenue_c.mean()
- rev_change = (rev_c / rev_control - 1) * 100
- Threshold: Must be > -2%
- Result: +5% (PASSED)

Check 3: Latency

- latency_c = 190ms
- Threshold: < 200ms
- Result: PASSED

Decision Matrix

Model	CTR	Rev & Lat	Sig?	Decision
Control	5.0%	Baseline	-	Baseline
CF (A)	6.1%	Pass	Yes	Consider
CB (B)	5.8%	Pass	No	Kill
Hybrid (C)	6.6%	Pass	Yes	SHIP

Final Recommendation

Ship Treatment C (Hybrid)

Rationale:

- Highest CTR: 6.6% (29% lift)
- Statistically significant ($p < 0.001$)
- Bayesian: 99.8% confidence
- All guardrails passed:
  - Revenue: +5% (> -2% threshold)
  - Latency: 190ms (< 200ms)
  - Error rate: 0.05% (< 0.1%)
- Expected annual value: $3.6M additional revenue

Alternative: Could also ship CF (A) as it also wins, but Hybrid is stronger

Task 6: Rollout Plan (5 minutes)

Gradual Rollout Strategy

Phase 1: Canary (Days 1-2)

- Deploy to 1% of users
- Monitor error rate, latency, revenue
- If stable after 24 hours $\rightarrow$ proceed
- Kill switch ready

Phase 2: Expansion (Days 3-5)

- 1% $\rightarrow$ 5% $\rightarrow$ 25%
- Each step: 24-hour monitoring
- Check guardrails at each step
- If any guardrail violated $\rightarrow$ rollback

Phase 3: Majority (Days 6-10)

- 25% $\rightarrow$ 50% $\rightarrow$ 100%
- Maintain 10% holdout for long-term measurement
- Monitor cohort retention over 30 days

Phase 4: Holdout Analysis (Days 11-40)

- Compare 90% treatment vs 10% control
- Measure 30-day retention, LTV

Monitoring Dashboard

Real-Time Metrics:

- CTR (hourly)
- Revenue per user (daily)
- Latency p50/p95/p99 (5-min intervals)
- Error rate (1-min intervals)
- Traffic distribution (control vs treatment)

Rollback Criteria

Immediate rollback if:

- Error rate $>$ 0.5% (5$\times$ baseline)
- Latency p95 $>$ 250ms (sustained 10 min)
- Revenue per user drops $>$ 10%
- User complaints spike $>$ 50/hour

Investigate if:

- CTR plateaus or decreases
- Revenue flat despite CTR increase
- Latency creeps above 200ms

Success Criteria

Workshop Takeaways & Course Conclusion

Week 10 Key Lessons

1. **Iteration is competitive advantage**
   - Winners ship 100+ experiments/year
   - Losers deploy once and hope
   - Velocity $\times$ win rate $=$ innovation speed
2. **Rigorous A/B testing prevents disasters**
   - Randomization enables causal inference
   - Statistical significance $\neq$ practical significance
   - Always include guardrails
3. **Bayesian methods accelerate learning**
   - $P(B > A)$ more intuitive than p-values
   - Earlier stopping possible
   - Incorporates prior knowledge
4. **Culture trumps tools**
   - Psychological safety for "failed" experiments
   - Data beats opinions
   - Fast decisions, slow reversions

Full Course Journey

10 Weeks, Complete ML Innovation Loop:

Empathize (Weeks 1-3):
- Clustering for user segmentation
- NLP for emotional context

Define (Week 4):
- Classification for problem framing

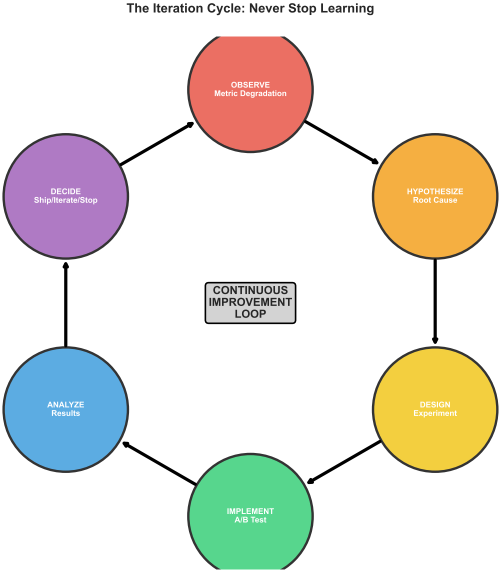
Ideate (Week 5):
- Topic modeling for idea generation

Prototype (Week 6):
- Generative AI for rapid prototyping

Test (Weeks 7-9):
- Responsible AI (ethics)
- Structured outputs (reliability)
- Multi-metric validation (rigor)

Iterate (Week 10):
- A/B testing for continuous improvement

The Continuous Improvement Loop: Never Stop Iterating



The 6-Stage Cycle

1. **Observe**
   - Monitor metrics
   - Listen to users
   - Watch competitors
2. **Hypothesize**
   - Form testable predictions
   - Identify risks
3. **Design**
   - Plan experiment
   - Calculate sample size
   - Define guardrails
4. **Implement**
   - Deploy test infrastructure
   - Monitor in real-time
5. **Analyze**
   - Run statistical tests
   - Check guardrails

Next Steps: Your A/B Testing Journey

Immediate Actions

1. Complete the Workshop (Today)

- Open the Jupyter notebook
- Run all 6 tasks end-to-end
- Document your decision rationale
- Share results with peers

2. Practice on Your Project (This Week)

- Identify ML model to test
- Design A/B test with guardrails
- Calculate required sample size
- Implement monitoring dashboard

3. Build Experimentation Culture (This Month)

- Run 1 experiment per week minimum
- Document learnings systematically
- Share results with stakeholders
- Celebrate "failed" experiments

Long-Term Mastery

4. Advanced Topics to Explore:

- Multi-armed bandits (adaptive allocation)
- Sequential testing (early stopping)
- Network effects & interference
- Long-term holdout analysis
- Variance reduction techniques
- Heterogeneous treatment effects

Resources

- Book: "Trustworthy Online Experiments" (Kohavi et al.)
- Tool: GrowthBook (open-source A/B platform)
- Course: Stanford CS329S (ML Systems Design)
- Community: Experiment Results Forum

Your Competitive Edge

You now know how to iterate 10× faster than peers who lack A/B testing skills. Use this advantage wisely.

Sustained experimentation drives innovation capacity - systematic testing infrastructure transforms how organizations evolve