

E-commerce Checkout Optimization: A/B Testing Analysis

Project Overview

This project analyzes a comprehensive A/B testing experiment conducted on an e-commerce platform to optimize the checkout process. The study examines the impact of design changes on conversion rates, revenue per user, and customer satisfaction across multiple user segments.

Business Problem

Challenge: The current checkout process has a 68% abandonment rate, costing the company an estimated \$2.3M annually in lost revenue.

Objective: Test a redesigned checkout flow to:

- Reduce checkout abandonment rate
- Increase overall conversion rate
- Improve average order value (AOV)
- Enhance user experience metrics

Success Metrics:

- Primary: Conversion rate (purchases/sessions)
- Secondary: Revenue per user, time to checkout completion
- Guardrail: User satisfaction score must not decrease

Experimental Design

Hypothesis

H₀: The new checkout design has no effect on conversion rate

H₁: The new checkout design increases conversion rate by at least 2%

Test Variants

- **Control (A):** Original 4-step checkout process
- **Treatment (B):** Streamlined 2-step checkout with progress indicators

Sample Size Calculation

python

```
import scipy.stats as stats
import numpy as np
from statsmodels.stats.power import ttest_power
from statsmodels.stats.proportion import proportions_ztest

# Power analysis for sample size
baseline_conversion = 0.124 # Current conversion rate
minimum_detectable_effect = 0.02 # 2% absolute increase
alpha = 0.05 # Significance level
power = 0.80 # Statistical power

# Calculate required sample size
effect_size = minimum_detectable_effect / np.sqrt(baseline_conversion * (1 - baseline_conversion))
sample_size_per_group = stats.norm.ppf(1 - alpha/2)**2 * 2 / effect_size**2
print(f"Required sample size per group: {int(sample_size_per_group):,}")
```

Sample Size: 15,680 users per group (31,360 total)

Test Duration: 21 days

Traffic Split: 50/50 randomized assignment

Randomization & Controls

- **Stratified randomization** by device type (mobile/desktop) and user type (new/returning)
- **Consistent user experience** - users remain in assigned group throughout test period
- **Exclusion criteria:** Users with incomplete session data, bot traffic, employees

Data Collection & Structure

Key Metrics Tracked

python

Primary metrics

```
conversion_rate = conversions / total_sessions
revenue_per_user = total_revenue / total_users
average_order_value = total_revenue / total_orders
```

Secondary metrics

```
time_to_checkout = checkout_completion_time - checkout_start_time
bounce_rate = single_page_sessions / total_sessions
cart_abandonment_rate = (carts_created - orders) / carts_created
```

Segmentation variables

```
device_type = ['mobile', 'desktop', 'tablet']
user_type = ['new_user', 'returning_user']
traffic_source = ['organic', 'paid', 'direct', 'social']
```

Data Quality Assurance

- **Sample Ratio Mismatch (SRM) Check:** Verified 50/50 split maintained
- **Outlier Detection:** Identified and handled extreme values (orders > \$5,000)
- **Data Validation:** Cross-referenced with payment processor data

Statistical Analysis

1. Primary Metric Analysis

Conversion Rate Results

python


```
# Conversion rate analysis
control_conversions = 1,847
control_sessions = 15,680
treatment_conversions = 2,156
treatment_sessions = 15,680

control_rate = control_conversions / control_sessions # 11.77%
treatment_rate = treatment_conversions / treatment_sessions # 13.75%

# Statistical significance test
z_stat, p_value = proportions_ztest([treatment_conversions, control_conversions],
                                     [treatment_sessions, control_sessions])

print(f"Control conversion rate: {control_rate:.3f}")
print(f"Treatment conversion rate: {treatment_rate:.3f}")
print(f"Relative lift: {(treatment_rate/control_rate - 1)*100:.1f}%")
print(f"p-value: {p_value:.4f}")
```

Results:

- **Control:** 11.77% conversion rate
- **Treatment:** 13.75% conversion rate
- **Absolute lift:** +1.98 percentage points
- **Relative lift:** +16.8%
- **Statistical significance:** $p < 0.001$ 

Revenue Impact Analysis

python

```
# Revenue per user analysis
```

```
control_rpu = 47.23
```

```
treatment_rpu = 52.91
```

```
# Bootstrap confidence intervals
```

```
def bootstrap_mean_diff(control_data, treatment_data, n_bootstrap=10000):
```

```
    bootstrap_diffs = []
```

```
    for _ in range(n_bootstrap):
```

```
        control_sample = np.random.choice(control_data, size=len(control_data), replace=True)
```

```
        treatment_sample = np.random.choice(treatment_data, size=len(treatment_data), replace=True)
```

```
        diff = np.mean(treatment_sample) - np.mean(control_sample)
```

```
        bootstrap_diffs.append(diff)
```

```
    return np.array(bootstrap_diffs)
```

```
# Calculate confidence interval for revenue lift
```

```
revenue_diff_ci = np.percentile(bootstrap_diffs, [2.5, 97.5])
```

```
print(f"Revenue per user lift: ${treatment_rpu - control_rpu:.2f}")
```

```
print(f"95% CI: [{revenue_diff_ci[0]:.2f}, {revenue_diff_ci[1]:.2f}]")
```

Revenue Results:

- **Control:** \$47.23 revenue per user
- **Treatment:** \$52.91 revenue per user
- **Absolute lift:** +\$5.68 per user (+12.0%)
- **95% CI:** [\$4.12, \$7.24]

2. Segmentation Analysis

Performance by Device Type

Segment	Control Conv Rate	Treatment Conv Rate	Lift	Significance
Mobile	9.2%	11.8%	+28.3%	p < 0.001 ✓
Desktop	15.1%	16.4%	+8.6%	p = 0.042 ✓
Tablet	11.7%	12.1%	+3.4%	p = 0.234 ✗

Performance by User Type

Segment	Control Conv Rate	Treatment Conv Rate	Lift	Significance
New Users	8.4%	10.9%	+29.8%	p < 0.001
Returning	16.2%	17.8%	+9.9%	p = 0.011

Key Insights:

- **Mobile users** show the largest improvement (+28.3% lift)
- **New users** benefit more than returning users
- Treatment is particularly effective for mobile-first customer segments

3. Secondary Metrics Analysis

User Experience Metrics

```
python

# Time to checkout analysis
control_avg_time = 284 # seconds
treatment_avg_time = 198 # seconds
time_reduction = ((control_avg_time - treatment_avg_time) / control_avg_time) * 100

print(f"Average checkout time reduction: {time_reduction:.1f}%")

# Cart abandonment analysis
control_abandonment = 0.731
treatment_abandonment = 0.654
abandonment_improvement = control_abandonment - treatment_abandonment

print(f"Cart abandonment rate improvement: -{abandonment_improvement:.3f}")
```

UX Results:

- **Checkout time:** 30.3% reduction (284s → 198s)
- **Cart abandonment:** 7.7 percentage point improvement
- **User satisfaction:** No significant change (guardrail metric protected)

4. Advanced Statistical Techniques

Bayesian Analysis

python

```
import pymc3 as pm
import arviz as az

# Bayesian A/B test analysis
with pm.Model() as model:
    # Priors
    p_control = pm.Beta('p_control', alpha=1, beta=1)
    p_treatment = pm.Beta('p_treatment', alpha=1, beta=1)

    # Likelihood
    obs_control = pm.Binomial('obs_control', n=control_sessions, p=p_control,
                              observed=control_conversions)
    obs_treatment = pm.Binomial('obs_treatment', n=treatment_sessions, p=p_treatment,
                                observed=treatment_conversions)

    # Derived quantities
    lift = pm.Deterministic('lift', p_treatment - p_control)
    relative_lift = pm.Deterministic('relative_lift', (p_treatment / p_control) - 1)

    # Sample from posterior
    trace = pm.sample(5000, tune=1000, return_inferencedata=True)

# Probability that treatment is better
prob_treatment_better = (trace.posterior.lift > 0).sum() / len(trace.posterior.lift.values.flat)
print(f"Probability treatment > control: {prob_treatment_better:.3f}")
```

Bayesian Results:

- **Probability treatment is better:** 99.7%
- **Expected lift:** 1.94% (95% HDI: [1.21%, 2.67%])
- **Risk of launching:** Very low

Multiple Testing Correction

python

```
from statsmodels.stats.multitest import multipletests

# Multiple hypothesis testing correction
p_values = [0.0003, 0.0089, 0.0421, 0.1847, 0.0156] # Various metric p-values
metric_names = ['Conversion Rate', 'Revenue per User', 'AOV', 'Bounce Rate', 'Time to Checkout']

# Benjamini-Hochberg correction
rejected, p_adjusted, _, _ = multipletests(p_values, alpha=0.05, method='fdr_bh')

for i, (metric, p_orig, p_adj, significant) in enumerate(zip(metric_names, p_values, p_adjusted, rejected)):
    print(f"{metric}: p={p_orig:.4f}, p_adj={p_adj:.4f}, Significant: {significant}")
```

Business Impact & ROI

Financial Impact Projection

python

```
# Annual revenue impact calculation
monthly_sessions = 125000
annual_sessions = monthly_sessions * 12

# Revenue impact
baseline_annual_revenue = annual_sessions * control_rate * 85.40 # avg order value
treatment_annual_revenue = annual_sessions * treatment_rate * 85.40

annual_revenue_lift = treatment_annual_revenue - baseline_annual_revenue
print(f"Projected annual revenue lift: ${annual_revenue_lift:,.0f}")

# Implementation costs
development_cost = 45000
ongoing_maintenance = 12000 # annual

roi = (annual_revenue_lift - ongoing_maintenance) / development_cost
print(f"First-year ROI: {roi:.1f}x")
```

Financial Results:

- **Projected annual revenue lift:** \$2,387,000
- **Implementation cost:** \$45,000

- **First-year ROI:** 52.1x
- **Payback period:** 0.7 months

Recommendation

✅ **LAUNCH RECOMMENDED**

Confidence Level: Very High

- Statistically significant results across primary metrics
- Consistent positive effects across key user segments
- Strong business case with excellent ROI
- Low risk based on Bayesian analysis

Implementation Strategy

Rollout Plan

1. **Week 1:** Deploy to 10% of traffic, monitor for technical issues
2. **Week 2:** Increase to 50% of traffic if no issues detected
3. **Week 3:** Full rollout to 100% of traffic
4. **Ongoing:** Monitor key metrics for 30 days post-launch

Success Monitoring

- **Daily monitoring** of conversion rates and revenue metrics
- **Weekly business reviews** with stakeholder updates
- **Monthly deep-dive analysis** including cohort analysis
- **Quarterly optimization** opportunities identification

Technical Implementation

Tools & Technologies

python

Statistical analysis stack

```
import pandas as pd
import numpy as np
import scipy.stats as stats
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
```

A/B testing specific libraries

```
from statsmodels.stats.power import ttest_power
from statsmodels.stats.proportion import proportions_ztest
import pymc3 as pm # Bayesian analysis
```

Data pipeline

```
import sqlalchemy
from airflow import DAG # Experiment monitoring automation
```

Experimental Infrastructure

- **Feature flagging:** LaunchDarkly for traffic allocation
- **Metrics collection:** Custom event tracking with Segment
- **Statistical analysis:** Python notebooks with automated reporting
- **Monitoring:** Real-time dashboards in Grafana

Key Learnings & Next Steps

Insights Gained

1. **Mobile optimization** shows disproportionate impact on conversion
2. **Simplified user flows** benefit new users significantly more
3. **Progress indicators** reduce anxiety and abandonment rates
4. **Checkout time reduction** correlates strongly with conversion improvement

Future Experiments

1. **Payment method optimization:** Test one-click payment options
2. **Trust signals:** Add security badges and customer reviews
3. **Personalization:** Dynamic checkout flow based on user behavior
4. **Mobile-first design:** Further mobile experience improvements

Methodology Improvements

1. Implement **sequential testing** for faster decision-making
2. Add **long-term cohort analysis** to measure retention impact
3. Develop **multi-armed bandit** approach for continuous optimization
4. Create **simulation framework** for pre-experiment power analysis

Skills Demonstrated

- ✓ **Experimental Design:** Hypothesis formulation, power analysis, randomization
 - ✓ **Statistical Analysis:** Frequentist and Bayesian methods, multiple testing correction
 - ✓ **Data Analysis:** Segmentation, outlier detection, confidence intervals
 - ✓ **Business Acumen:** ROI calculation, risk assessment, implementation planning
 - ✓ **Technical Implementation:** Python, SQL, statistical libraries
 - ✓ **Communication:** Clear reporting, stakeholder presentations, actionable insights
 - ✓ **Product Thinking:** User experience focus, iterative improvement
-

Project Duration: 6 weeks (3 weeks experiment + 3 weeks analysis)

Sample Size: 31,360 users

Statistical Significance: $p < 0.001$

Business Impact: \$2.4M annual revenue lift

ROI: 52.1x first-year return

This project demonstrates comprehensive A/B testing expertise with rigorous statistical methods and clear business impact measurement.