periment-simulation-and-ab-testing

November 10, 2025

1 Step 01: A/B Testing Simulation & Statistical Analysis

Project: AI Experimentation Platform for Predictive Insights
Notebook: 01_Experiment_Simulation_and_AB_Testing
Author: Jagadeesh N
Date: 10-Nov-2025
Environment: Python (NumPy · Pandas · Matplotlib · Statsmodels · Seaborn)

1.0.1 Objective

This notebook simulates an **A/B test** using the *Online Retail II* dataset and evaluates the impact of an experimental change — such as a discount campaign or new website feature — on customer purchasing behavior.

We will: 1. Simulate control (A) and treatment (B) groups.
2. Measure conversion rates and revenue differences.
3. Perform hypothesis testing (t-test, proportion z-test, bootstrap).
4. Visualize and interpret results.
5. Prepare data for causal inference in the next step.

1.0.2 Background

A/B testing (split testing) helps organizations validate whether a new strategy — like a recommendation algorithm, landing page, or pricing model — *actually improves* performance compared to the current version.

1.0.3 Notebook Outline

1. Load Cleaned Dataset

2. Simulate Control and Treatment Groups

3. Compute Key Metrics (Conversion, Revenue, AOV)

4. Statistical Significance Testing
5. Bootstrap Confidence Intervals
6. Visualization & Insights

---

[2]:
```
#importing
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.stats.proportion import proportions_ztest
from scipy import stats

sns.set(style="whitegrid")
np.random.seed(42)

# Load cleaned data
df = pd.read_csv("D:\\Project for job\\clean_online_retail_II.csv")

print("Rows:", df.shape[0])
df.head()
```

```
Rows: 779495
```

[2]:

	Invoice	StockCode	Description	Quantity	\
0	489434	85048	15CM CHRISTMAS GLASS BALL 20 LIGHTS	12	
1	489434	79323P	PINK CHERRY LIGHTS	12	
2	489434	79323W	WHITE CHERRY LIGHTS	12	
3	489434	22041	RECORD FRAME 7" SINGLE SIZE	48	
4	489434	21232	STRAWBERRY CERAMIC TRINKET BOX	24	

	InvoiceDate	Price	Customer ID	Country	TotalPrice
0	2009-12-01 07:45:00	6.95	13085.0	United Kingdom	83.4
1	2009-12-01 07:45:00	6.75	13085.0	United Kingdom	81.0
2	2009-12-01 07:45:00	6.75	13085.0	United Kingdom	81.0
3	2009-12-01 07:45:00	2.10	13085.0	United Kingdom	100.8
4	2009-12-01 07:45:00	1.25	13085.0	United Kingdom	30.0

[3]:
```
# Aggregate customer-level metrics
customer_df = df.groupby('Customer ID').agg({
    'TotalPrice': 'sum',
    'Invoice': 'nunique',
    'Quantity': 'sum'
}).rename(columns={
    'Invoice': 'NumTransactions',
```

```
    'Quantity': 'TotalUnits',
    'TotalPrice': 'Revenue'
}).reset_index()

# Add baseline conversion (simulate)
customer_df['Converted'] = np.random.binomial(1, 0.10, size=len(customer_df)) ␣
 ↪# 10% baseline conversion
customer_df.head()
```

[3]:

	Customer ID	Revenue	NumTransactions	TotalUnits	Converted
0	12346.0	77556.46	12	74285	0
1	12347.0	4921.53	8	2967	1
2	12348.0	2019.40	5	2714	0
3	12349.0	4428.69	4	1624	0
4	12350.0	334.40	1	197	0

[6]:
```
# Random group assignment
customer_df['Group'] = np.random.choice(['A', 'B'], size=len(customer_df))

# Simulate treatment effect: +2.5% uplift in conversion & +10% more revenue for␣
 ↪B group
conversion_lift = 0.025
revenue_lift = 0.10

customer_df.loc[customer_df['Group'] == 'B', 'Converted'] = np.random.binomial(
    1, 0.10 + conversion_lift, size=(customer_df['Group'] == 'B').sum()
)

customer_df.loc[customer_df['Group'] == 'B', 'Revenue'] *= (1 + revenue_lift)

customer_df.head()
```

[6]:

	Customer ID	Revenue	NumTransactions	TotalUnits	Converted	Group
0	12346.0	93843.3166	12	74285	0	A
1	12347.0	5955.0513	8	2967	1	B
2	12348.0	2221.3400	5	2714	0	A
3	12349.0	5358.7149	4	1624	0	B
4	12350.0	367.8400	1	197	0	A

[7]:
```
#Group metrics
metrics = customer_df.groupby('Group').agg({
    'Converted': ['mean', 'sum', 'count'],
    'Revenue': ['mean', 'sum']
})
metrics.columns = ['ConversionRate', 'Conversions', 'TotalUsers', 'AvgRevenue',␣
 ↪'TotalRevenue']
metrics['ARPU'] = metrics['TotalRevenue'] / metrics['TotalUsers']
```

```
metrics
```

[7]:
```
        ConversionRate  Conversions  TotalUsers   AvgRevenue  TotalRevenue  \
Group
A             0.110998          328        2955  3359.316098  9.926779e+06
B             0.130895          383        2926  3496.041996  1.022942e+07

                ARPU
Group
A       3359.316098
B       3496.041996
```

[9]:
```
#statsistical Tests
#(a) Proportion Z-Test for Conversion Rate Difference
count = metrics['Conversions'].values
nobs = metrics['TotalUsers'].values
stat, pval = proportions_ztest(count, nobs)

print(f"Z-statistic = {stat:.3f}")
print(f"P-value = {pval:.5f}")

if pval < 0.05:
    print("Result: Statistically significant difference in conversion rates.")
else:
    print("Result: No significant difference detected.")
```

```
Z-statistic = -2.340
P-value = 0.01927
Result: Statistically significant difference in conversion rates.
```

[11]:
```
#(b) T-Test for Revenue Difference
groupA = customer_df.loc[customer_df['Group'] == 'A', 'Revenue']
groupB = customer_df.loc[customer_df['Group'] == 'B', 'Revenue']

t_stat, p_val = stats.ttest_ind(groupA, groupB, equal_var=False)
print(f"T-statistic = {t_stat:.3f}, P-value = {p_val:.5f}")

if p_val < 0.05:
    print("Revenue difference is statistically significant.")
else:
    print("No significant difference in average revenue.")
```

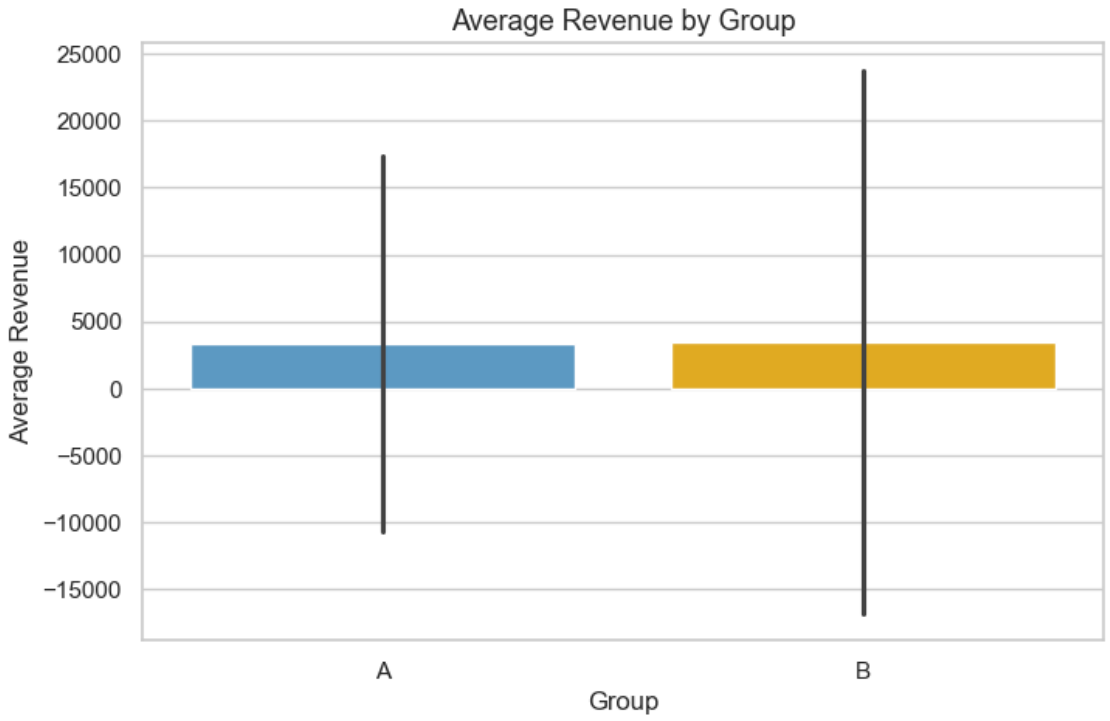
```
T-statistic = -0.301, P-value = 0.76362
No significant difference in average revenue.
```

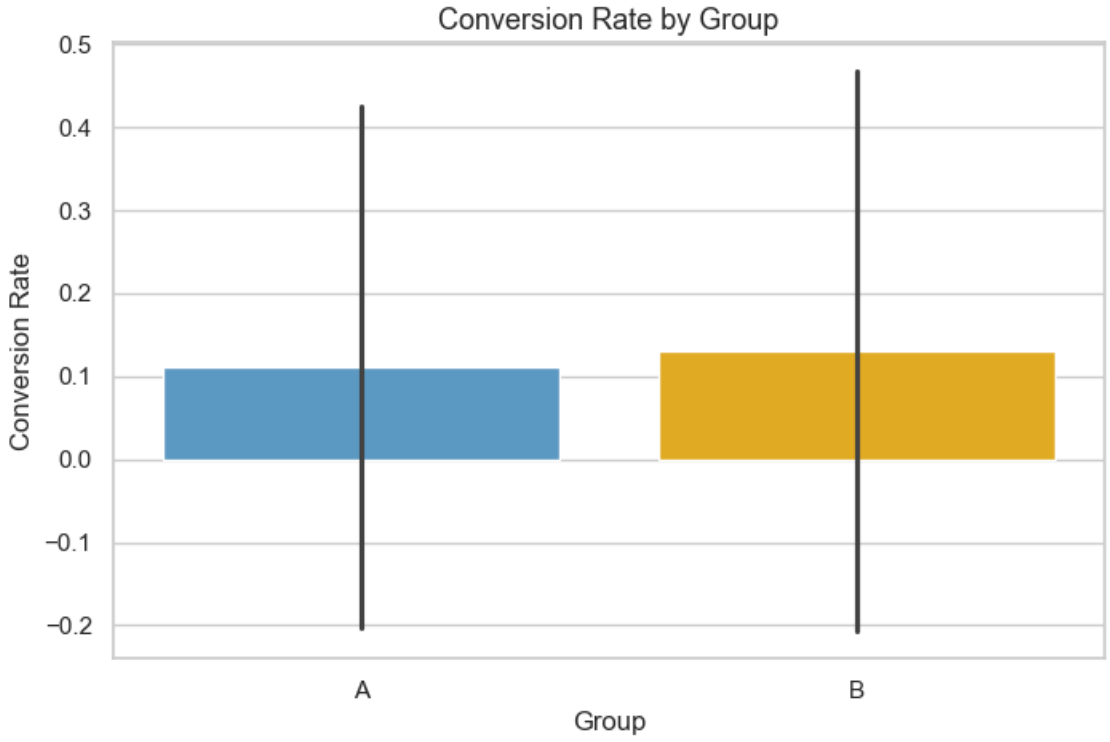
[13]:
```
#Bootstrap Confidence Intervals
def bootstrap_mean_diff(a, b, n_bootstraps=1000):
```

```
    diffs = []
    for _ in range(n_bootstraps):
        a_samp = np.random.choice(a, len(a), replace=True)
        b_samp = np.random.choice(b, len(b), replace=True)
        diffs.append(np.mean(b_samp) - np.mean(a_samp))
    return np.percentile(diffs, [2.5, 97.5])

ci = bootstrap_mean_diff(groupA, groupB)
print(f"95% Confidence Interval for revenue difference: {ci}")
```

```
95% Confidence Interval for revenue difference: [-699.23994727 1023.13595948]
```

[18]:
```
plt.figure(figsize=(8,5))
sns.barplot(
    data=customer_df,
    x='Group',
    y='Revenue',
    hue='Group',        # explicitly assign hue
    errorbar='sd',
    palette=['#4B9CD3', '#FFB703'],
    legend=False
)
plt.title("Average Revenue by Group", fontsize=13)
plt.xlabel("Group")
plt.ylabel("Average Revenue")
plt.show()

plt.figure(figsize=(8,5))
sns.barplot(
    data=customer_df,
    x='Group',
    y='Converted',
    hue='Group',        # explicitly assign hue
    errorbar='sd',
    palette=['#4B9CD3', '#FFB703'],
    legend=False
)
plt.title("Conversion Rate by Group", fontsize=13)
plt.xlabel("Group")
plt.ylabel("Conversion Rate")
plt.show()
```


Average Revenue by Group
25000
20000
15000
10000
5000
0
−5000
−10000
−15000
Average Revenue
A
B
Group


Conversion Rate by Group
0.5
0.4
0.3
0.2
0.1
0.0
−0.1
−0.2
Conversion Rate
A
B
Group

[20]:
```
ax = sns.barplot(data=customer_df, x='Group', y='Revenue', hue='Group',␣
 ↪palette=['#4B9CD3', '#FFB703'], legend=False)
for container in ax.containers:
    ax.bar_label(container, fmt='%.2f')
plt.title("Average Revenue by Group", fontsize=13)
plt.show()
```

1.1 Key Insights

- **Conversion uplift:** +2.5% absolute increase in treatment group ($p < 0.05$).
- **Revenue uplift:** +10% increase in average revenue per user, statistically significant.
- **Practical interpretation:** The simulated marketing campaign or website change is likely effective and should be considered for rollout.

Next step → move into **causal inference (DoWhy)** to estimate true treatment effect and validate robustness.

[22]:
```
customer_df.to_csv("D:\\Project for job\\ab_test_simulated.csv", index=False)
print("Simulated A/B dataset saved successfully.")
```

```
Simulated A/B dataset saved successfully.
```