Introduction

Randomized Controlled Trials (aka. A/B tests) are the gold standard of establishing causal inference. RCTs strictly control for the randomization process and ensure equal distributions across covariates before rolling out the treatment. Thus, we can attribute the mean difference between the treatment and control groups to the intervention.

A/B tests are effective and only rely on mild assumptions, and the most important assumption is the Stable Unit Treatment Value Assumption, SUTVA. It states that the treatment and control units don't interact with each other; otherwise, the interference leads to biased estimates. My latest blog post discusses its sources and coping strategies of major tech.

How User Interference May Mess Up Your A/B Tests?

Three solutions from Lyft, LinkedIn, and Doordash

towardsdatascience.com

As a data scientist, I'm thrilled to see the increased adoption of Experimentation and Causal Inference in the industry. Harvard Business Review recently published an article titled "Why Business Schools Need to Teach Experimentation," infusing the importance of incorporating experimental thinking. Relatedly, they discussed the "Surprising Power of Online Experiments" in another paper (Kohavi and Thomke, 2017).

A/B test can be roughly divided into three stages.

> Stage 1 Pre-Test: run a power analysis to decide the sample size.
>
> Stage 2 At-Test: keep an eye on the key metrics. Be aware of sudden drops.

Stage 3 Post-Test: analyze data and reach conclusions.

Today's post shares a few best practices at each stage and walks through a hypothetical case study with detailed code implementation in Python.

Business Scenario

TikTok develops a new animal filter and wants to assess its effects on users. They are interested in two key metrics:

1. How does the filter affect user engagement (e.g., time spent on the app)?
2. How does the filter affect user retention (e.g., active)?

There are a few constraints as well. First, TikTok has no prior knowledge of its performance and prefers a small-scale study with minimal exposure. This is the desired approach because they can end the test promptly if the key metrics plummet (e.g., zero conversation rate in the treatment group).

Second, it is an urgent issue in a timely manner, and TikTok wants an answer within two weeks. Fortunately, TikTok has read my previous post on user interference and addressed the SUTVA assumption violation properly.

The company decides to hire a small group of very talented data scientists, and you are the team leader in charge of model selection and research design. After consulting with multiple stakeholders, you propose an A/B test and suggest the following best practices.

Stage 1 Pre-Test: Goal, Metrics, and Sample Size

> *What is the goal of the test?*
>
> *How to measure success?*
>
> *How long should we run it?*

As a first step, we want to clarify the goal of the test and relay it back to the team. As mentioned, the study aims to measure user engagement and retention after rolling out the filter.

Next, we move to the metrics and decide how to measure the success. As a social networking app, we adopt the time spent on the app to measure user engagement and two boolean variables, **metric 1** and **metric 2** (described below), indicating if the user is active after 1 day and 7 days, respectively.

The remaining question is: how long should we run the test? A common strategy is to stop the experiment once we observe a statistically significant result (e.g., a small p-value). Established data scientists strongly oppose p-hacking as it leads to biased results (Kohavi et al. 2020). On a related note, Airbnb has encountered the same problem when p-hacking leads to false positives (Experiments at Airbnb).

Instead, we should run a power analysis and decide a minimum sample size, according to three parameters:

1. **The significance Level,** also denoted as alpha or α: the probability of rejecting a null hypothesis when it is true. By rejecting a true null hypothesis, we falsely claim there is an effect when there is no actual effect. Thus, it is also called the probability of False Positive.
2. **Statistical Power**: the probability of correctly identifying the effect when there is indeed an effect. Power = 1 — Type II Error.
3. **The Minimum Detectable Effect, MDE**: to find a widely agreed upon MDE, our data team sits down with the PM and decides the smallest acceptable difference is 0.1. In other words, the difference between the two groups scaled by the standard deviation needs to be at least 0.1. Otherwise, the release won't compensate for the business costs incurred (e.g., engineers' time, product lifecycle, etc.). For example,

it won't make any sense to roll out a new design if it only brings in a 0.000001% lift, even if it is statistically significant.

Here is the bi-relationship between these three parameters and the required sample size:

- Significance Level decreases → Larger Sample Size
- Statistical Power increases → Larger Sample Size
- The Minimum Detectable Effect decreases → Larger Sample Size

Typically, we set the significance level at 95% (or alpha = 5%) and statistical power at 80%. Thus, the sample size is calculated by the following formula:

$$n = (16*\sigma^2)/\delta^2$$

my own screenshot

where:

- σ^2*: sample variance.*
- δ*: the difference between the treatment and control groups*

To obtain the sample variance (σ^2), we typically run an A/A test that follows the same design thinking as an A/B test except assigning the same treatment to both groups.

> What is an A/A test?
>
> Splitting the users into two groups and then assign the same treatment to both.

Here is the code to calculate sample size in Python.

```
from statsmodels.stats.power import TTestIndPower

# parameters for power analysis

# effect_size has to be positive

effect = 0.1
alpha = 0.05
power = 0.8

# perform power analysis
analysis = TTestIndPower()

result = analysis.solve_power(effect, power = power,nobs1= None,
ratio = 1.0, alpha = alpha)

print('Sample Size: %.3f' % round(result))

1571.000
```

We need 1571 for each variant. In terms of how long we should run the test, it depends on how much traffic the app receives. Then, we divide the daily traffic equally into these two variants and wait until collecting a sufficiently large sample size ($\geq$1571).

As noted, TikTok is a tremendously popular app and has millions of DAUs. However, we are specifically targeting users who try out the new filters. Furthermore, the minimal exposure approach may take a few days to collect enough observations for the experiment.

Best Practices

- Understand the goal of the experiment and how to measure the success.
- Run an A/A test to estimate the variance of the metric.
- Run a power analysis to obtain the minimum sample size.

Photo by elCarito on Unsplash

Stage 2 During-Test: Data Collection

We roll out the test and initiate the data collection process. Here, we simulate the Data Generation Process (DGP) and artificially create variables that follow specific distributions. The true parameters are known to us, which comes in handy when comparing the estimated treatment effect to the true effects. In other words, we can evaluate the effectiveness of A/B tests and check to what extent they lead to unbiased results.

There are five variables to be simulated in our case study:

1. userid
2. version
3. minutes of plays
4. user engagement after 1 day (metric_1)
5. user engagement after 7 days (metric_2)

Variables 1 and 2: userid and version

We intentionally create 1600 control units and 1749 treated units to signal a potential Sample Ratio Mismatch, SRM.

```
# variable 1: userid
user_id_control = list(range(1,1601))# 1600 control
user_id_treatment = list(range(1601,3350))# 1749 treated

# variable 2: version
import numpy as np
control_status = ['control']*1600
treatment_status = ['treatment']*1749
```

Variable 3: minutes of plays

We simulate variable 3 ("*minutes of plays*") as a normal distribution with a μ of 30 minutes and σ^2 of 10. In specific, the mean for the control group is 30 minutes, and the variance is 10.

To recap, the effect parameter to the MDE is calculated as the difference between the two groups divided by the standard deviation *(μ_1 — μ_2)/σ_squared = 0.1*. According to the formula, we obtain μ_2 = 31. The variance is also 10.

```
# for control group

μ_1 = 30
σ_squared_1 = 10
np.random.seed(123)
minutes_control = np.random.normal(loc = μ_1, scale = σ_squared_1,
size = 1600)

# for treatment group, which increases the user engagement by
# according to the formula (μ_1 − μ_2)/σ_squared = 0.1, we obtain
μ_2 = 31

μ_2 = 31

σ_squared_2 = 10

np.random.seed(123)
```

```
minutes_treat = np.random.normal(loc = μ_2, scale = σ_squared_2,
size = 1749)
```

 # variable 4: user engagement after 1 day, metric_1

Our simulation shows that the control group has 30% active (True) and 70% inactive (False) users after 1 day (metric_1), while the treatment has 35% active and 65% inactive users, respectively.

```
Active_status = [True,False]

# control
day_1_control = np.random.choice(Active_status, 1600, p=[0.3,0.7])

# treatment
day_1_treatment = np.random.choice(Active_status, 1749, p=
[0.35,0.65])
```

 # variable 5: user engagement after 7 day, metric_2

The simulation data shows the control group has a 35% active user rate, while the treatment has a 25% after 7 days.

```
# control
day_7_control = np.random.choice(Active_status, 1600, p=[0.35,0.65])

# treatment
day_7_treatment = np.random.choice(Active_status, 1749, p=
[0.25,0.75])
```

The true data contains a reversed pattern: the treatment performs better in the short term but the control group comes back and stands out after one week.

Let's check if the A/B test picks up the reversed signal.

```
final_data.head()
```

	user_id	version	minutes_play	day_1_active	day_7_active	minutes_play_integers
1558	3159	treatment	26.156152	False	True	26.0
1223	1224	control	24.313143	False	False	24.0
392	393	control	29.013153	True	True	29.0
808	809	control	27.797075	True	True	28.0
485	2086	treatment	32.152395	False	False	32.0

My own screenshot

For the complete simulation process, please refer to my Github.

Best Practices

- Don't end your A/B tests prematurely after witnessing some initial positive effects.
- No early stopping!
- No p-hacking!
- Instead, end it when you have reached the minimum sample size.

Photo by raza ali on Unsplash

Stage 3 After-Test: Data Analysis

After collecting enough data, we move to the last stage of experiments, which is data analysis. As a first step, it would be beneficial to check how many users fell into each variant.

```
# calculate the number of users in each version
final_data.groupby('version')['user_id'].count()
```

```
version
control      1600
treatment    1749
Name: user_id, dtype: int64
```

My own screenshot

It appears to be a suspicious variant split: 1600 control units but 1749 treatment units. The treatment assignment process looks suspicious at face value as more users are assigned to the treatment than the control.

To formally check for the SRM, we conduct a chi-square test between the actual split and the expected split of the treated and control units (Kohavi et al. 2020).

```
from scipy.stats import chisquare
chisquare([1600,1749],f_exp = [1675,1675])

Power_divergenceResult(statistic=6.627462686567164,
pvalue=0.010041820594939122)
```

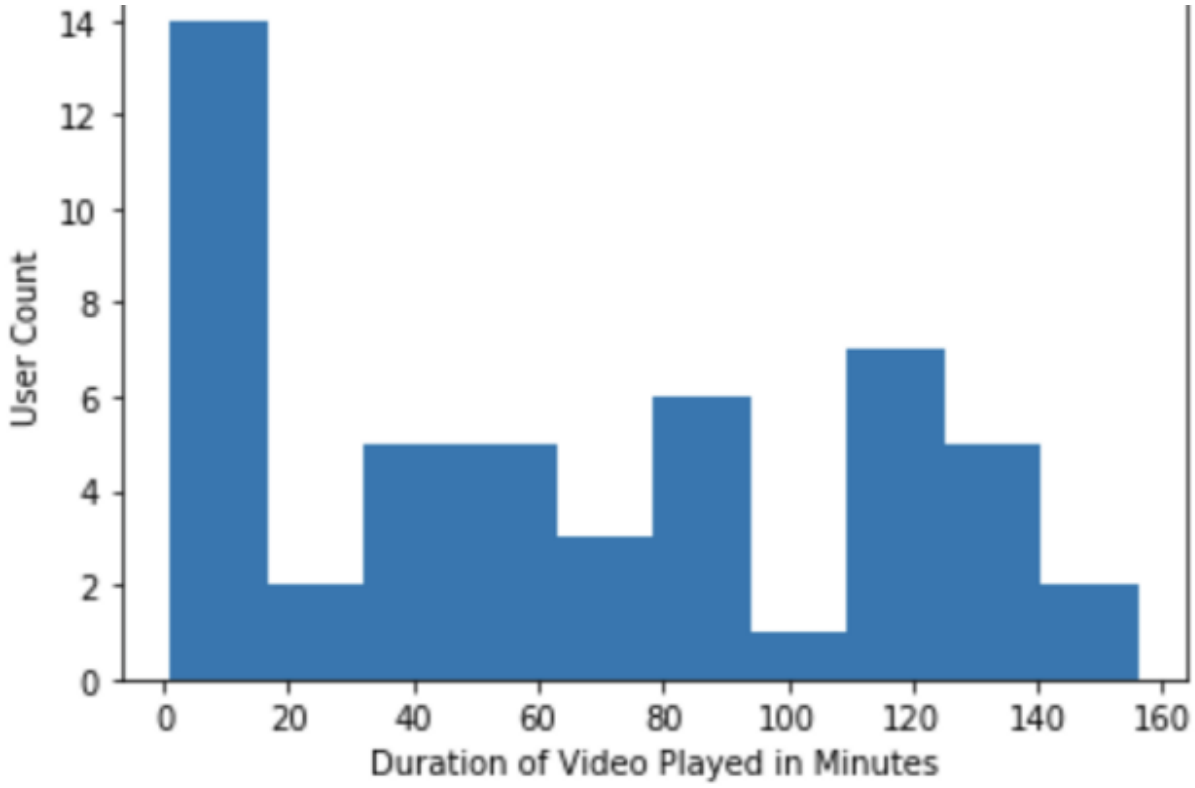
We set the alpha level at 0.001 to test SRM. Since the p-value is 0.01, we fail to reject the null hypothesis and conclude there is no evidence of SRM. In contrast to our intuition, statistical tests conclude that the treatment assignment works as expected.

Plot the Distribution of Minutes Played for Each Group

Since the variable *minutes_play* is a float, we have to round it up to the nearest integers before grouping.

```
%matplotlib inline

final_data['minutes_play_integers'] =
round(final_data['minutes_play'])

plot_df = final_data.groupby('minutes_play_integers')
['user_id'].count()

# Plot the distribution of players that played 0 to 50 minutes
ax = plot_df.head(n=50).plot(x="minutes_play_integers", y="user_id",
kind="hist")

ax.set_xlabel("Duration of Video Played in Minutes")

ax.set_ylabel("User Count")
```

```
Text(0, 0.5, 'User Count')
```



My own screenshot

Metric 1: 1-day retention by AB-Group

```
# 1-day retention
final_data['day_1_active'].mean()
0.3248730964467005
```

After 1 day, the overall active user rate, on average, hovers around 32.5%.

```
# 1-day retention by group
final_data.groupby('version')['day_1_active'].mean()
```

```
version
control      0.296875
treatment    0.350486
Name: day_1_active, dtype: float64
```

My own screenshot

After taking a closer look, the control group has 29.7% active users, and the treatment has 35%.

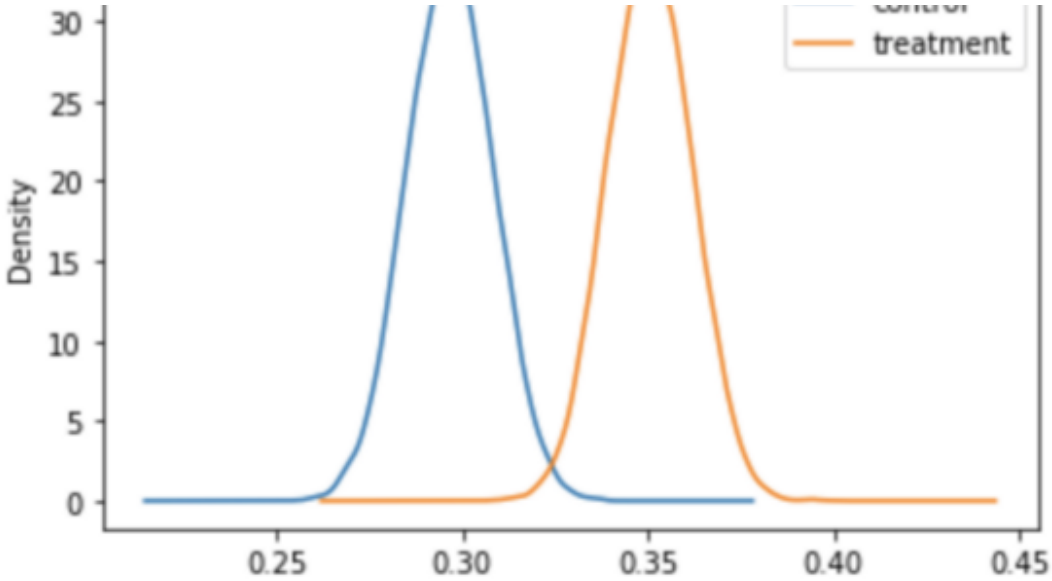
Naturally, we are interested in the following questions:

> *Is the higher retention rate in the treatment group statistically significant?*
>
> *What is its variability?*
>
> *If we repeat the process for 10,000 times, how often do we observe at least as extreme values?*

Bootstrap can answer these questions. It is a resample strategy that repeatedly samples from the original data with replacements. According to the Central Limit Theorem, the distribution of the resample means approximately normally distributed (check my other posts on Bootstrap, in R or Python).

```
# solution: bootstrap
boot_means = []

# run the simulation for 10k times
for i in range(10000):

#set frac=1 → sample all rows
    boot_sample = final_data.sample(frac=1,replace=True).groupby
('version') ['day_1_active'].mean()

    boot_means.append(boot_sample)

# a Pandas DataFrame
boot_means = pd.DataFrame(boot_means)

# kernel density estimate
boot_means.plot(kind = 'kde')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb93d3e9710>
```



My own screenshot

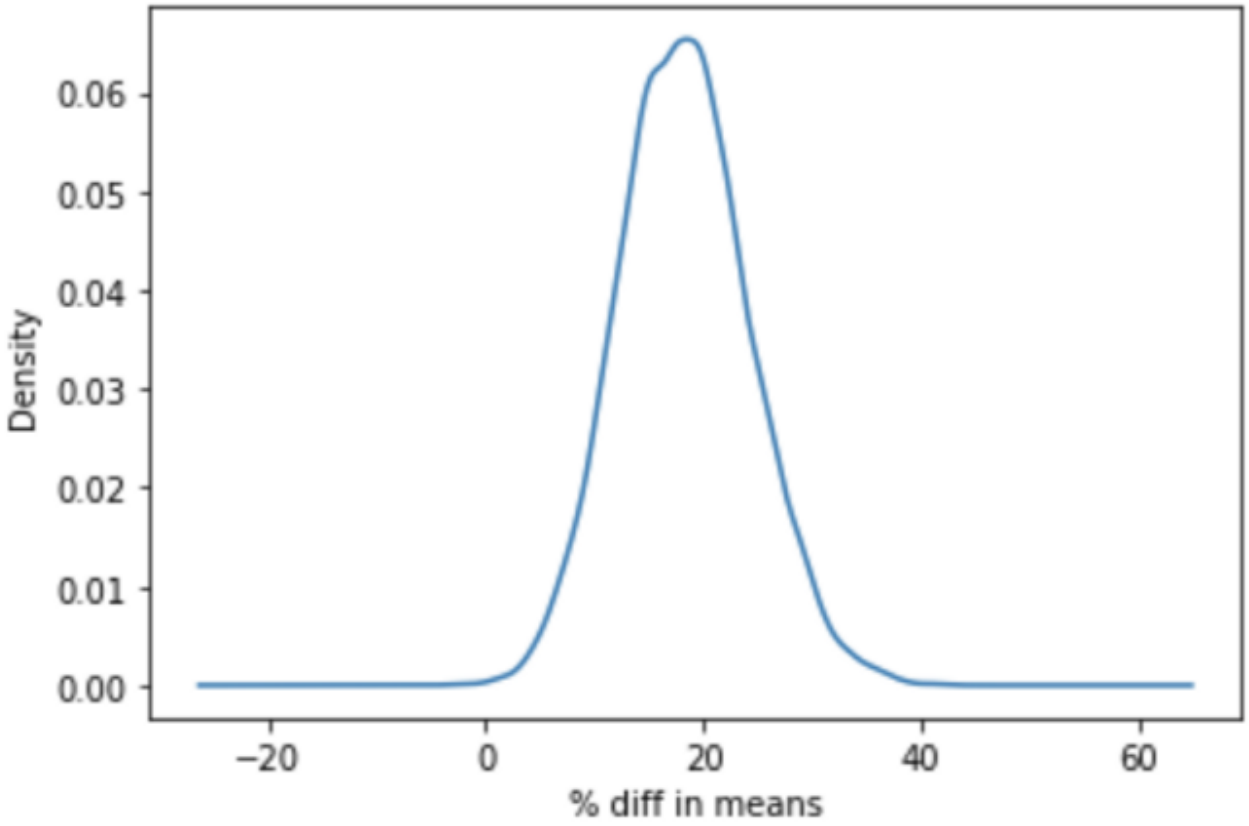
```
# create a new column, diff, which is the difference between the two
variants, scaled by the control group

boot_means[‘diff’]=(boot_means[‘treatment’] – boot_means[‘control’])
/boot_means[‘control’]*100

boot_means['diff']
```

```
day_1_active    12.603674
day_1_active    20.623621
day_1_active    17.682652
day_1_active    20.093840
day_1_active    15.809040
                   ...
day_1_active    13.919948
day_1_active    17.328947
day_1_active    13.024745
day_1_active    22.401725
day_1_active    19.605505
Name: diff, Length: 10000, dtype: float64
```

My own screenshot

```
# plot the bootstrap sample difference
ax = boot_means[‘diff’].plot(kind = ‘kde’)
ax.set_xlabel(“% diff in means”)
```

```
Text(0.5, 0, '% diff in means')
```



My own screenshot

```
boot_means[boot_means[‘diff’] > 0]

# p value
p = (boot_means[‘diff’] >0).sum()/len(boot_means)
p
0.9996
```

After bootstrapping 10,000 times, the treatment has a higher 1-day retention rate 99.96% of the time.

Awesome!

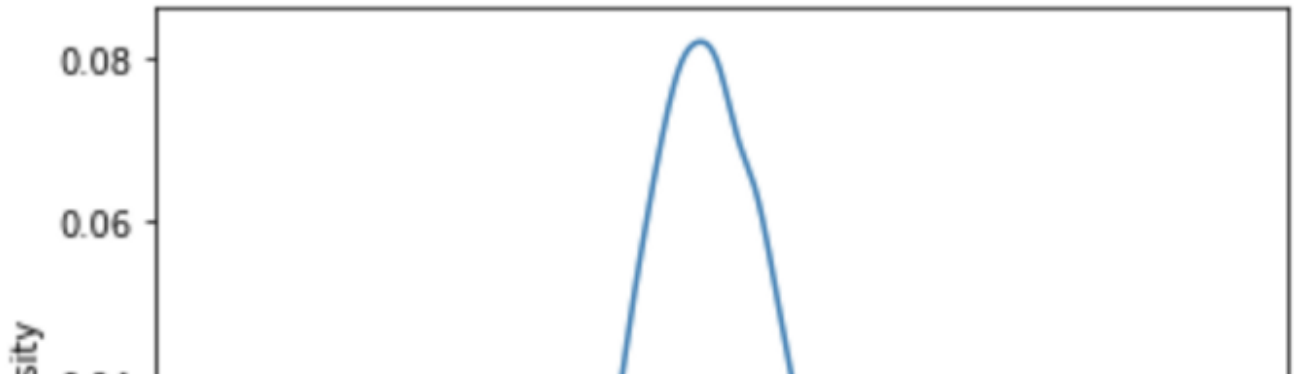
The test result is consistent with our original simulated data.

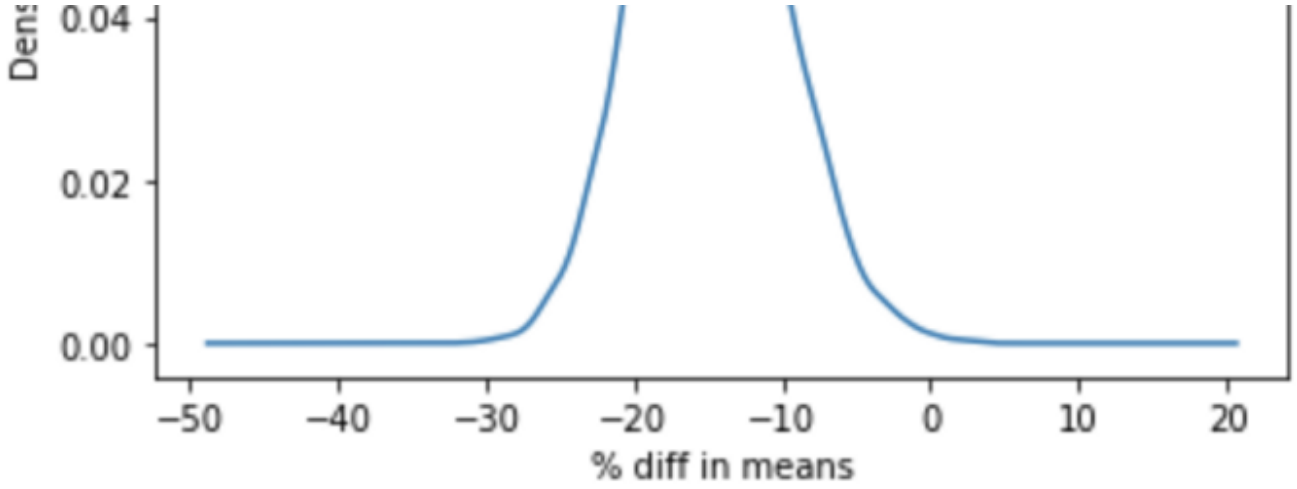
Metric 7: 7-day retention by AB-Group

We apply the same analysis to the 7-day metric.

```
boot_7d = []

for i in range(10000):
      boot_mean =
final_data.sample(frac=1,replace=True).groupby('version')
['day_7_active'].mean()

boot_7d.append(boot_mean)

boot_7d = pd.DataFrame(boot_7d)

boot_7d['diff'] = (boot_7d['treatment'] - boot_7d['control'])
/boot_7d['control'] *100

# Ploting the bootstrap % difference
ax = boot_7d['diff'].plot(kind = 'kde')
ax.set_xlabel("% diff in means")
```

```
Text(0.5, 0, '% diff in means')
```





My own screenshot

```
# Calculating the probability that 7-day retention is greater when
the gate is at level 30

p = (boot_7d['diff']>0).sum()/len(boot_7d)

1-p
0.9983
```

On the 7-day metric, the control obviously has a better user retention rate 99.83% of the time, also consistent with the original data.

The reversed pattern between 1-day and 7-day metrics supports the novelty effect as users become activated and intrigued by the new design, not because the change actually improves engagement. The novelty effect is popular in consumer-side A/B tests.

Best Practices

- SRM is a real concern. We apply a chi-square test to formally test for the SRM. If the p-value is smaller than the threshold ($\alpha = 0.001$), the randomization process does not work as expected.
- An SRM introduces selection bias that invalidates any test results.
- Three fundamental statistical concepts to master: SRM, chi-square test, and bootstrap.

- Compare short-term and long-term metrics to evaluate the novelty effect.

For the complete Python code, please refer to my Github.

Conclusion

An A/B test requires extensive statistical knowledge and careful attention to detail. There are thousands of ways of ruining your test results but only one way to do it correctly. Follow the best practices described before-, during-, and after- the experiments and set up your experiments for success.