Code Applied

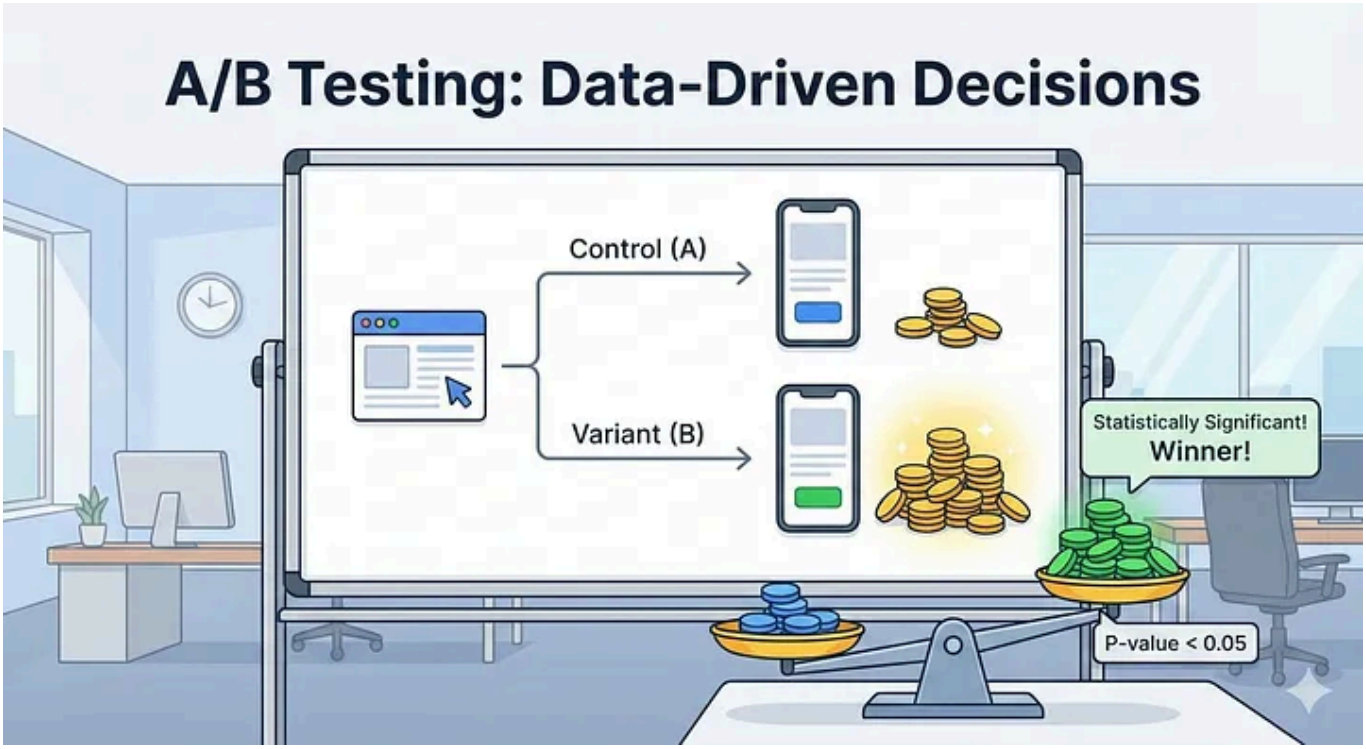
Member-only story

# A Beginner's Guide to A/B Testing: Tips, Tricks, and Best Practices

Gustavo R Santos 6 min read · 19 hours ago



Stop Guessing. The Data Science Secret to Scaling Your Business with A/B Testing




A/B Testing. | Image generated by AI. Google, 2026. https://gemini.google.com

A/B Testing is more than the classic discussion of what sell more: the blue or the red *Buy* button. Maybe you thought that “Electric Lime” would drive even more sales, while people insisted on “Classic Blue.”

But, well, we should stop guessing and start experimenting more. **A/B testing** is the referee you’ve always needed. It’s a simple experiment where we show two versions of something to different users. Then, you see which one performs better.

## A or B: Does It Matter?

In the digital world, tiny changes lead to big profits. A/B testing lets you make data-backed decisions. Instead of relying on the loudest person in the room, you rely on your customers’ actual behavior.

It’s basically the scientific method for your business. You form a hypothesis, run an experiment, and analyze the results. It’s clean, it’s logical, and it saves

you from making expensive mistakes.

A/B Tests are backed by statistical hypothesis tests, where we compare two groups to see if they are really different from each other, given a confidence interval.

You don't need a math degree, but you do need to understand "significance." Essentially, we want to know if our results happened because of our change or just by pure luck.

So, we use a "P-value" to measure this. If your P-value is less than 0.05, it means there's less than a 5% chance the result was a fluke. That's usually the "go" signal.

## Statistical Power

Because we are testing whether the intervention works, we want strong evidence before concluding that it is effective. This is where **statistical power** comes in.

*Power = 1 − Type II error*

Out of all possible outcomes (100%), we subtract the probability of failing to detect a true treatment effect.

- Power represents the probability that we **do not miss a real effect**.
- The higher the power, the more confident we can be that our test will identify an effective intervention when it truly works.
- It is like the brightness of a flashlight you're using to find a tiny key in a dark room. If the light's too dim, you might miss the key entirely. In A/B

testing, that “key” is the boost in sales or clicks you’re looking for.

> Power is the probability that your test will actually spot that boost if it’s really there.

#### Why 80% is the Magic Number

In the industry, we usually aim for **80% power**. Why not 100%? Well, to be 100% sure, you’d need an infinite number of users, which nobody has time for.

At 80%, you’re accepting a 20% risk of missing a winner. It’s a fair trade-off between being accurate and actually getting your product out the door.

## Code

In the Python script we will go over in this example, we used `proportions_ztest`. It’s a handy tool from the `statsmodels` library that compares the conversion rates of two different groups.

1. The code takes the number of conversions. We can consider a conversion as our *success*.
2. Take the total number of visitors. Each visitor to the website can be considered an *attempt*.
3. Once we perform the test, it spits out a P-value that tells us if one proportion is statistically different than the other. In other words, *is the test group different than the control group?*

Let’s import the modules.

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

To implement the sample size calculation based on a power of 80%, here is an approach.

- We will consider 5% significance level.
- A Power of 80%, which is the industry standard.
- By setting your **Power** to 0.80 and your **Alpha** to 0.05, you're telling the script: *"I want to be 95% sure I'm not seeing a fake win, and 80% sure I don't miss a real one."*
- Our base conversion is 10%.
- The minimum effect we expect to see is 1% increase, otherwise it won't worth the value.
- We then use the `power.NormalIndPower()` method, since our test will be a proportions Z Test.

```
# Settings for the test
alpha = 0.05      # Significance level (5% risk of a false positive)
power_val = 0.80  # Statistical Power (80% chance of detecting a real effect)
base_rate = 0.1  # Your current conversion rate (10%)
minimum_expected_effect = 0.01 # The minimum increase expected (1%)

# Calculate 'Effect Size' (how big the change is relative to noise)
# We use a simple Cohen's h approximation or standardized difference
effect_size = minimum_expected_effect / base_rate

# Initialize the power analysis object
analysis = power.NormalIndPower()
```

```
# Solve for the required number of observations per group
required_n = analysis.solve_power(
    effect_size=effect_size, # Standardized effect size
    alpha=alpha,
    power=power_val,
    ratio=1.0 # Equal size for both groups
)

print(f"You need approximately {int(round(required_n))} visitors per group.")
print(f"Total traffic needed: {int(round(required_n)) * 2}")
```

```
You need approximately 1570 visitors per group.
Total traffic needed: 3140
```

Great. We already know our sample size now. We need each group with at least 1570 visitors.

### The A/B Test

Now, we simulate some data. Let's say that our first group of 1600 visitors

```
# Simulation data:
# Group A (Control): 1000 visitors, 120 conversions
# Group B (Variant): 1050 visitors, 155 conversions
conversions = np.array([120, 155])
visitors = np.array([1600, 1650])
```

And perform the test.

```
# We perform a two-sided Z-test
# Null Hypothesis: The conversion rates are the same.
stat, p_value = proportions_ztest(conversions, visitors)
```

```
# Calculate conversion rates for display
rate_a = conversions[0] / visitors[0]
rate_b = conversions[1] / visitors[1]

print(f"Control Conversion Rate: {rate_a:.2%}")
print(f"Variant Conversion Rate: {rate_b:.2%}")
print(f"P-Value: {p_value:.4f}")

# Check for significance (alpha = 0.05)
if p_value < 0.05:
    print("Result is statistically significant! Deploy the variant.")
else:
    print("Not enough evidence. Keep testing or stick with the control.")
```

```
Control Conversion Rate: 7.50%
Variant Conversion Rate: 9.39%
P-Value: 0.0524
Not enough evidence. Keep testing or stick with the control.
```

Simple and effective, huh?!

## Real-Life Applications

Imagine you run an e-commerce site. You're wondering if a "Buy Now" button works better than "Add to Cart."

You split your traffic 50/50. After a week, you let the data tell the story.

Maybe "Buy Now" increased sales by 15%. Without testing, you'd never know for sure. Companies like Netflix and Amazon run thousands of these tests simultaneously to perfect their user experience.

## Avoid the "Peeking" Trap

One big mistake people make is checking results too early. It's tempting to stop the test as soon as you see a winner. But statistics need a full sample size to be valid.

If you stop early, you might catch a temporary "winner" that eventually loses. It's called the *Peeking Problem*.

Pick a sample size beforehand and stick to it!

## The Novelty Effect

Sometimes users click something new just because it's new. This is the *Novelty Effect*. Your click rate might spike for three days and then crash back to normal.

To beat this, run your tests for at least one or two full business cycles. Usually, two weeks is a solid baseline for most websites. This helps clear out any temporary excitement.

## Segmentation is Key

Your "Electric Lime" button might flop with older users, but thrive with Gen Z. Don't just look at the total average. Break your data down by device, location, or age.

Segmenting your results can reveal hidden gems. A "losing" test overall might actually be a huge win for your mobile users. Always dig a little deeper into the demographics.

## Before You Go

A/B testing turns your business into a learning machine. It replaces ego with evidence. Start small, test often, and let the data lead the way to your next big win.

Ready to see if that new change in your website actually works?

Go experiment with an A/B Test and let your users decide!

## References


**statsmodels.stats.proportion.proportions_ztest - statsmodels 0.14.6**

Edit description

www.statsmodels.org

**Statistical functions (scipy.stats) - SciPy v1.16.2 Manual**

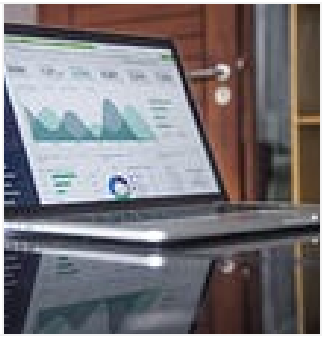
This module contains a large number of probability distributions, summary and frequency statistics, correlation...

docs.scipy.org

**A gentle introduction to the mathematics behind A/B testing | Towards Data Science**

A/B testing is a tool that allows to check whether certain causal relationship hold. For example, a data scientist...

towardsdatascience.com



**statsmodels.stats.power.NormalIndPower.power - statsmodels 0.14.6**

Edit description

www.statsmodels.org

**A/B Testing using Python - GeeksforGeeks**

Your All-in-One Learning Portal: GeeksforGeeks is a comprehensive educational platform that empowers learners across...

www.geeksforgeeks.org

Statistics | Data Science | A B Testing | Python | Experimentation


## Published in Code Applied

157 followers · Last published 19 hours ago

Following

Code Applied delivers practical, bite-sized tutorials on data science, AI agents, automation, and more. Each post packs real code, clear insights, and weekend-worthy experiments to level up your skills. Learn fast. Build smart. Apply what matters.

## Written by Gustavo R Santos

3.5K followers · 37 following

Edit profile

Data Scientist | I solve business challenges through the power of data. | Visit my site: https://gustavorsantos.me

## Responses (1)

Gustavo R Santos

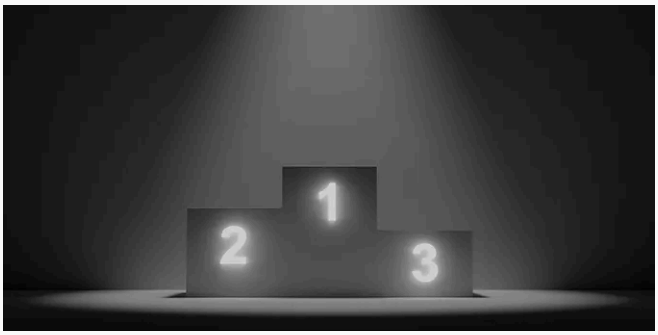
What are your thoughts?

gab1930s
19 hours ago

Go experiment with an A/B Test and let your users decide!

Reply

## More from Gustavo R Santos and Code Applied




In TDS Archive by Gustavo R Santos

### Creating Scores and Rankings with PCA

Use R Language to create scores for observations based on many variables

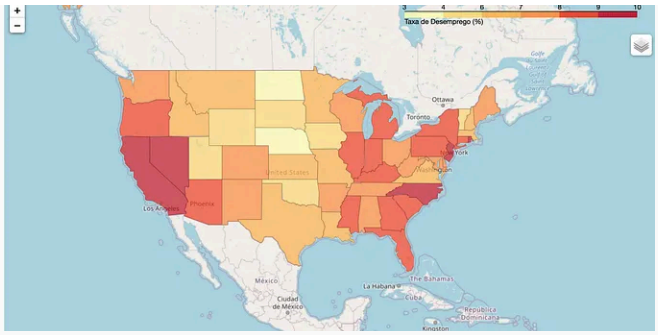
Apr 10, 2023 21


In Code Applied by Gustavo R Santos

### 5 Useful Visualizations to Enhance Your Analysis

Use Python's statistical visualization library Seaborn to level up your analysis.

Dec 18, 2025 64


In Code Applied by Gustavo R Santos

## Documenting Python Projects with MkDocs

Use Markdown to quickly create a beautiful documentation page for your projects

Dec 20, 2025 54

In Data Hackers by Gustavo R Santos

## Criando Mapas Interativos e Choropleth Maps com Folium em...

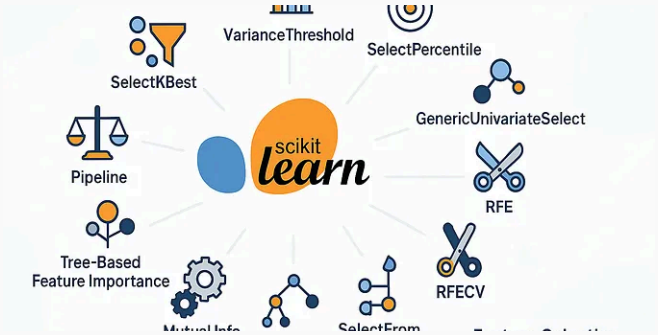
Aprenda a criar mapas simples e a adicionar codificação de valores baseada em cores...

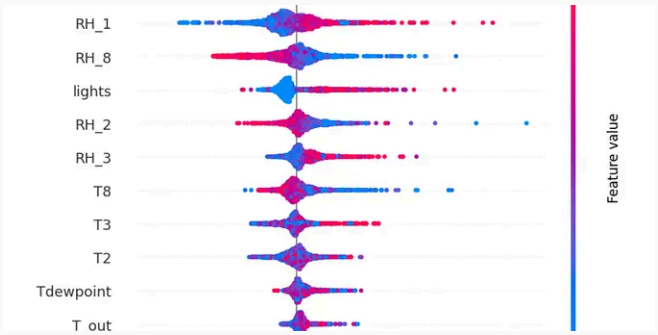
Aug 31, 2020 77 2

See all from Gustavo R Santos

See all from Code Applied

# Recommended from Medium



Bhagya Rana

## 10 Feature Selection Techniques Built into Scikit-learn That Every...

Master the built-in tools Scikit-learn offers for smarter, faster, and more interpretable...

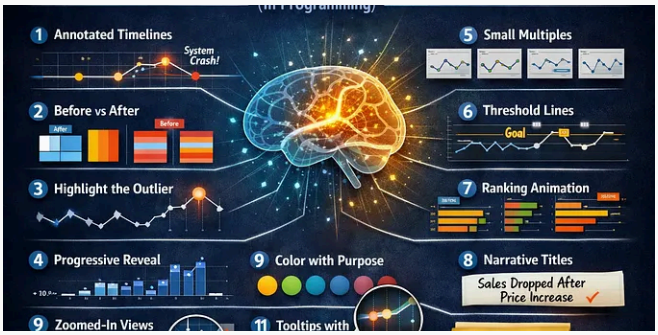
Aug 5, 2025 38 1

DropletOfData

## From Correlation to Causation: Modeling Appliance Energy...

Most energy forecasting systems optimize for accuracy and stop there. That's a problem.

Dec 23, 2025 14

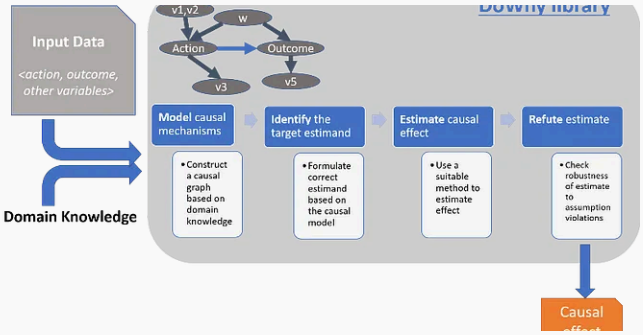



Code Crush

## 12 Visualization Hacks That Turn Data Into Stories

Great data visualization is not about charts. It's about control.

Dec 29, 2025 8

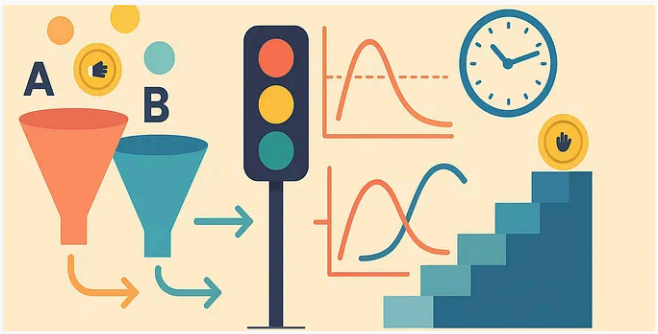



Eugene Zinoviev

## Causal Inference in Decision Intelligence—Part 9: DoWhy...

Comparing two causal inference frameworks offered by the DoWhy library.

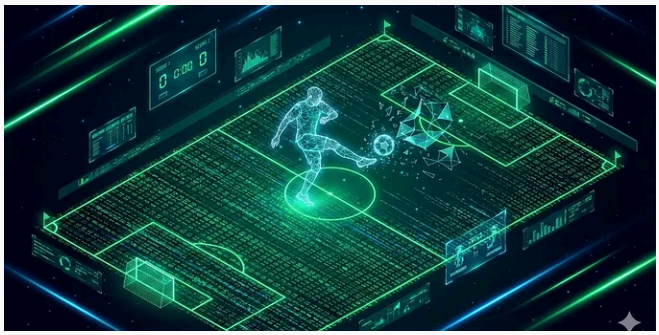
Sep 7, 2025 23




Codastra

## Probability, Patience, and Peeking in A/B Tests

A practical guide to power, sample size, and why early looks can sink your p-values—and...



Arjun Kaarat

## The Ludic Fallacy of the Perfect Captaincy: A Data Scientist's Guid...

Why Monte Carlo simulations, Portfolio Theory, and the "Eye Test" couldn't save me...

Oct 26, 2025 31 5d ago

See more recommendations