The Winner's Curse

Significance Testing is Not an Endorsement of the Treatment Effect

Russ Zaliznyak <rzaliznyak@gmail.com>

2025-12-01

Table of contents

Introduction

Significance Testing is being vastly misused in industry. There are two key flaws:

1. Significance Testing is too risk averse in business contexts. It requires too much evidence and winners are passed aside in favor of statistical purity.
2. Significance Testing is incorrectly being used as an endorsement of the Treatment Effect observed in the experiment.

In this paper we address only **#2** — namely that *Significance Testing is not an endorsement of the Treatment Effect*. But we do offer a *Bayesian* method to improve effect size estimates.

AB Test Setup

To prove our claim, we're going to design a properly-powered *Significance Test*.

Baseline Metric

```
# Define Baseline Conversion Rate
control_mean = 0.65
```

Runtime & Allocation

```
max_test_n = int(132_000)
control_traffic_share = 0.50
```

Significance Parameters

```
alpha = 0.05
minimum_detectable_effect = 1.01
```

AB Test Design

Let's draw the *Null Hypothesis* and *Alternative Hypothesis* below.

Null Hypothesis

```
from numpy.random import binomial
from numpy import percentile, concatenate, interp, asarray, array, mean, sort, searchsorted
import plotly.figure_factory as ff
import plotly.graph_objects as go

NUMBER_SIMULATIONS = int(1e6)

control_trials = int(
    max_test_n * control_traffic_share
)
treatment_trials = (
    max_test_n - control_trials
)
control_posterior_samples = binomial(
    n=control_trials,
    p=control_mean,
    size=NUMBER_SIMULATIONS,
)/ control_trials

control_posterior_samples_1 = binomial(
    n=treatment_trials,
    p=control_mean,
    size=NUMBER_SIMULATIONS,
)/treatment_trials


difference_of_means = (
    control_posterior_samples_1
    - control_posterior_samples
)


significance_threshold = percentile(difference_of_means, 100 - alpha*100)
fig = ff.create_distplot(
    [difference_of_means],
    group_labels=["Null Distribution"],
    show_hist=False,
    show_rug=False,
)
fig.data[0].showlegend = False
# Convert tuple traces to NumPy arrays so boolean masking works
x = asarray(fig.data[0].x, dtype=float)
y = asarray(fig.data[0].y, dtype=float)


sorted_vals = sort(difference_of_means)
n = len(sorted_vals)

# Vectorized empirical CDF
cdf_vals = searchsorted(sorted_vals, x, side="right") / n

fig.data[0].customdata = cdf_vals
fig.data[0].hovertemplate = (
    "x = %{x:.4f}<br>"
    "P(X < x) = %{customdata:.2%}<extra></extra>"
)

# Right-tail mask
mask = x >= significance_threshold
if mask.any():
    x_tail = x[mask]
    y_tail = y[mask]

    # Interpolate y at the exact threshold to make a clean polygon
    y_thresh = interp(significance_threshold, x, y)

    # Closed polygon for shaded area
    poly_x = concatenate(
        ([significance_threshold], x_tail, [x_tail[-1], significance_threshold])
    )
    poly_y = concatenate(([y_thresh], y_tail, [0], [0]))

    fig.add_trace(
        go.Scatter(
            x=poly_x,
            y=poly_y,
            fill="toself",
            mode="lines",
            line=dict(width=0),
            fillcolor="rgba(255,0,0,0.3)",
            name="<b>StatsSig</b>",
        )
    )
    fig.add_trace(
    go.Scatter(
        hoverinfo="skip",
        # name=" ",
        name="avg",
        x=[mean(difference_of_means)],
        y=[0.000],
        mode="markers",
        marker_symbol="circle",
        showlegend=False,
        marker=dict(
            color="red",
            size=20,
            line=dict(color="black", width=3),
        ),
    )
)

# Vertical reference line
fig.add_vline(
    x=significance_threshold,
    line_dash="dash",
    line_color="red",
    annotation_text=f"95th pct = {significance_threshold:.2%}",
    annotation_position="top right",
)

fig.update_layout(
    title="Difference of Means - Right Tail Shaded (  95th percentile)",
    xaxis_tickformat=".1%",
    yaxis=dict(
        showticklabels=False,   # hides tick labels
        ticks="",               # hides the tick marks
    ),
    height = 400
)
fig.show()
```

```
Unable to display output for mime type(s): text/html
```

```
Unable to display output for mime type(s): text/html
```

Alternative Hypothesis