A/B testing

STATISTICAL THINKING IN PYTHON (PART 2)

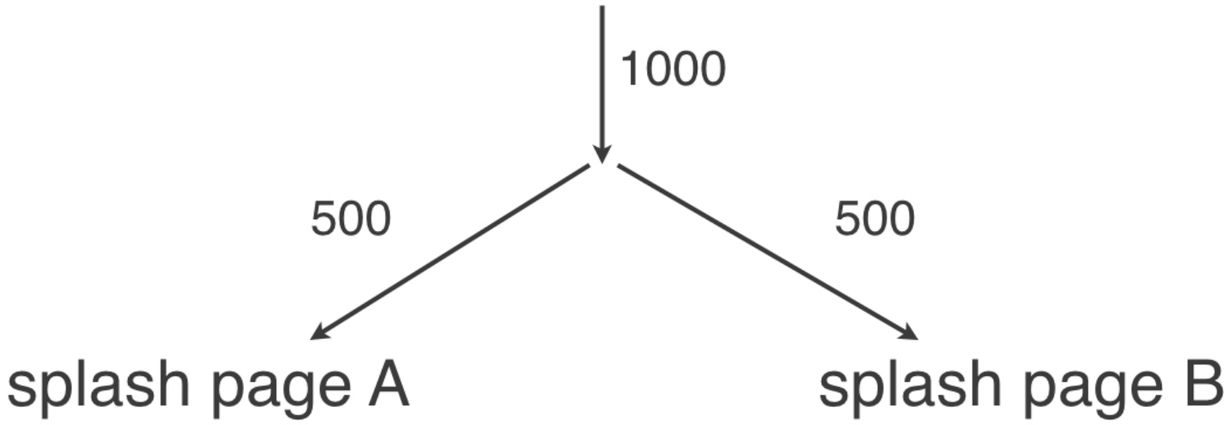
Justin Bois

Lecturer at the California Institute of Technology

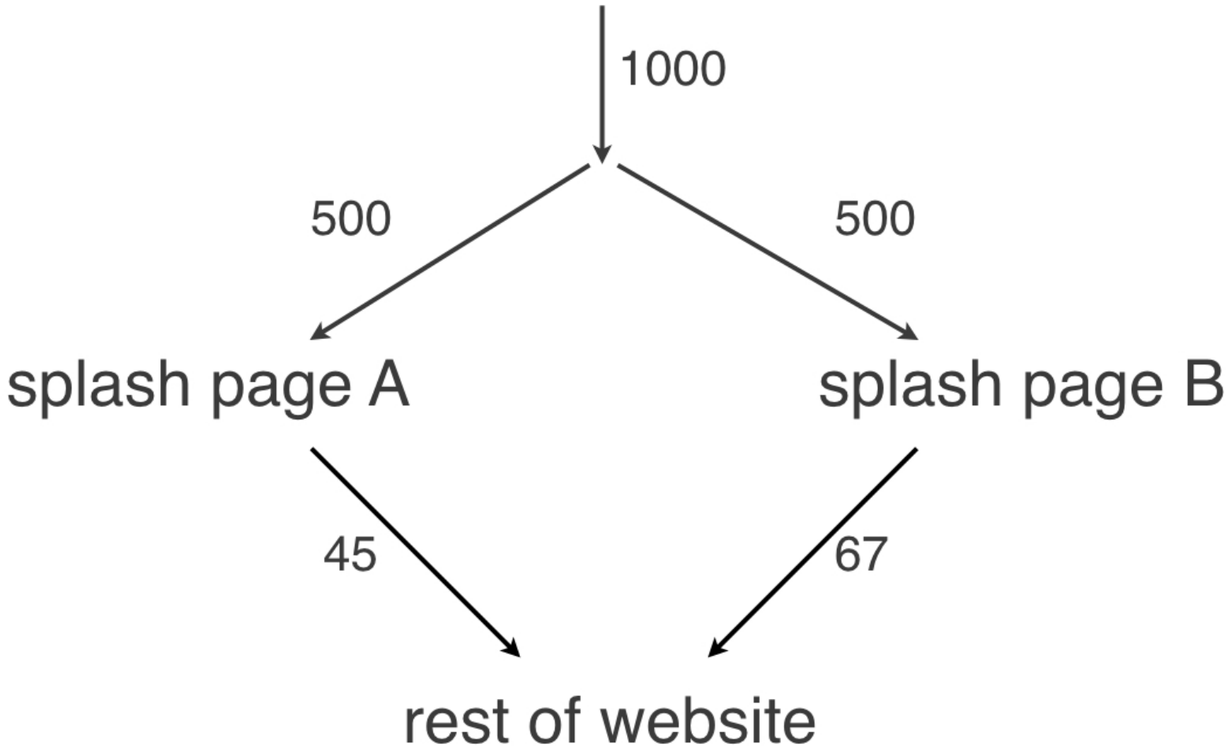
datacamp

Is your redesign effective?

Take a set of 1000 visitors to the site and direct 500 of them to the original splash page and 500 of them to the redesigned one. You determine whether or not each of them clicks through to the rest of the website



Is your redesign effective?



On the original page, which we'll call page A, 45 visitors clicked through, and on the redesigned page, page B, 67 visitors clicked through. This makes you happy because that is almost a 50% increase in the click-through rate. But maybe there really is no difference between the effect of two designs on click-through rate and the difference you saw is due the random chance.

You want to check: what is the probability that you would observe at least the observed difference in number of clicks through if that were the case? This is asking exactly the question you can address with a hypothesis test.

A permutation test is a good choice here because you can simulate the result as if the redesign had no effect on the click-through rate.

Null hypothesis

- The click-through rate is not affected by the redesign

Permutation test of clicks through

```
import numpy as np
# clickthrough_A, clickthrough_B: arr. of 1s and 0s
def diff_frac(data_A, data_B):
        frac_A = np.sum(data_A) / len(data_A)
        frac_B = np.sum(data_B) / len(data_B)
        return frac_B - frac_A
diff_frac_obs = diff_frac(clickthrough_A,
                          clickthrough_B)
```

for each splash page design, we have a Numpy array which contains 1 or 0 values for whether or not a visitor clicked through. Next, we need to define a function diff_frac for our test statistic. Ours is the fraction of visitors who click through. We can compute the fraction who click through by summing the entries in the arrays of ones and zeros and then dividing by the number of entries. Finally we compute the observed value of the test statistic using this function diff_frac.

Permutation test of clicks through

```
perm_replicates = np.empty(10000)
for i in range(10000):
     perm_replicates[i] = permutation_replicate(
                clickthrough_A, clickthrough_B, diff_frac)
p_value = np.sum(perm_replicates >= diff_frac_obs) / 10000
p_value
```

```
0.016
```

Now everything is in place to generate our permutation replicates of the test statistic

using the permutation_replicate function you wrote in the exercises; we will generate 10,000. We compute the p-value as the number of replicates where the test statistic was at least as great as what we observed. We get a value of 0-point-016, which is relatively small, so we might reasonably think that the redesign is a real improvement. This is an example of an A/B test.

A/B test

- Used by organizations to see if a strategy change gives a better result

Null hypothesis of an A/B test

- The test statistic is impervious to the change

Let's practice!

STATISTICAL THINKING IN PYTHON (PART 2)

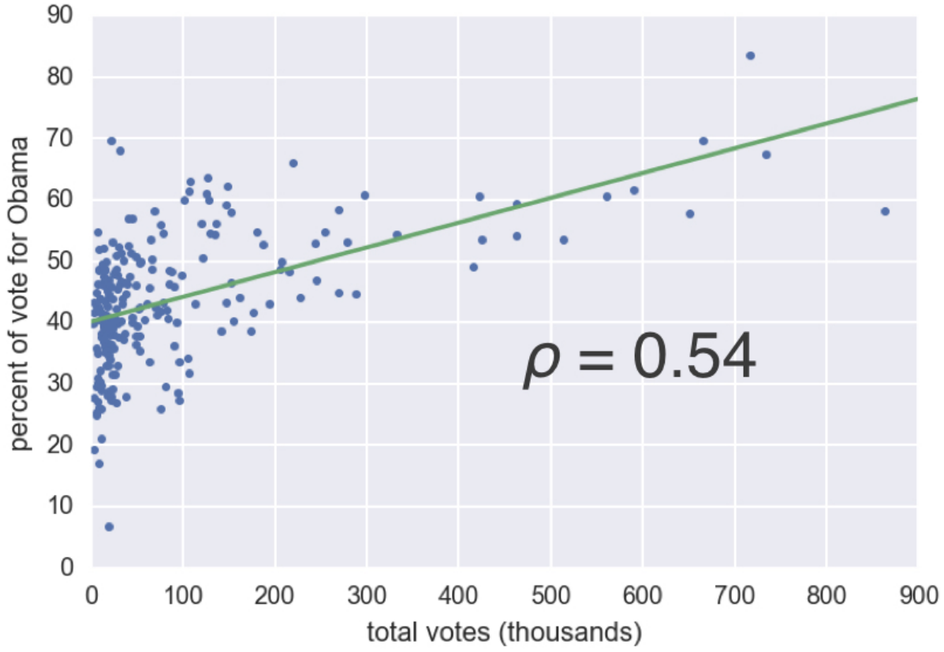
Test of correlation

STATISTICAL THINKING IN PYTHON (PART 2)

Justin Bois

Lecturer at the California Institute of Technology

2008 US swing state election results

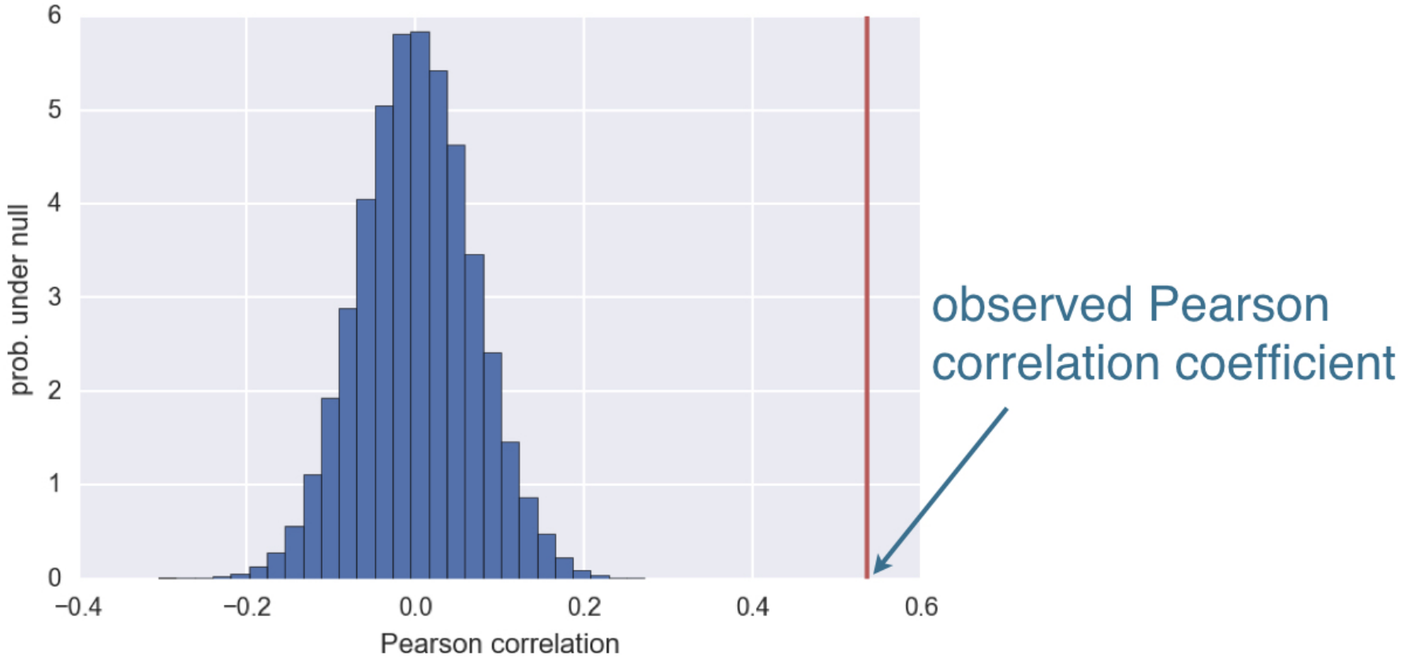


[1] Data retrieved from Data.gov (https://www.data.gov/)

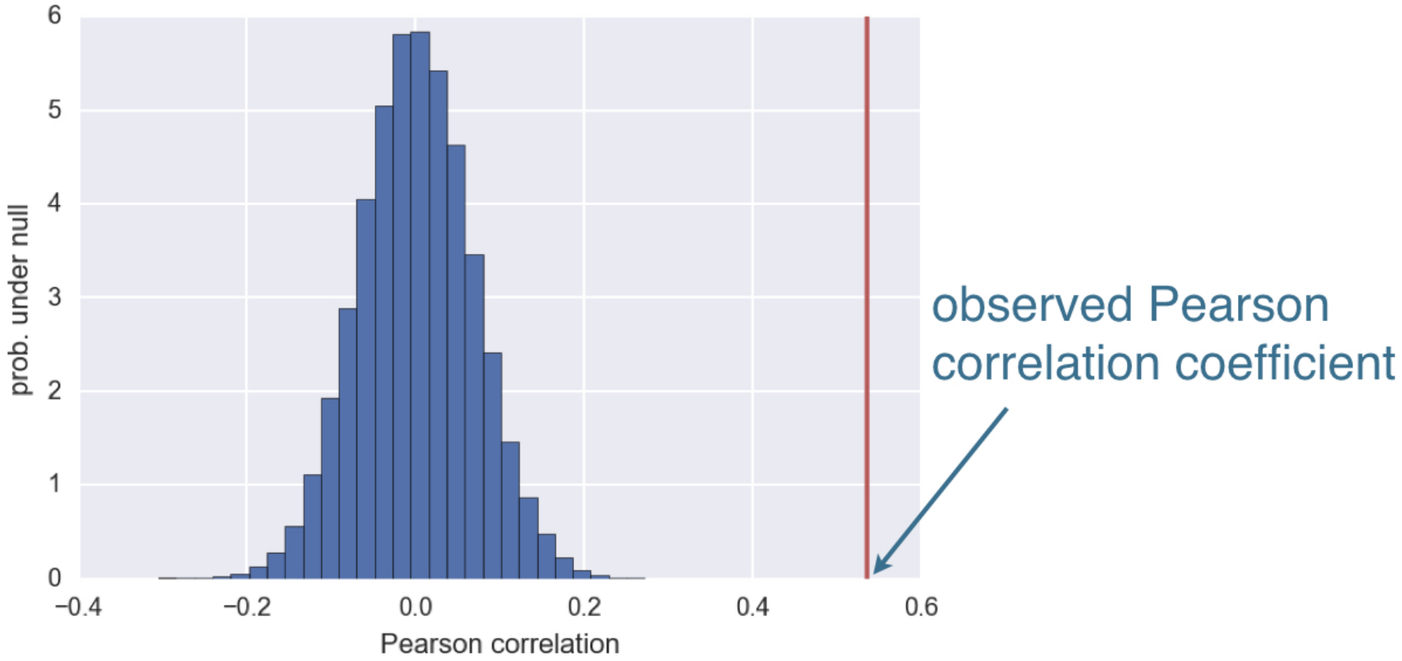
Hypothesis test of correlation

- Posit null hypothesis: the two variables are completely uncorrelated
- Simulate data assuming null hypothesis is true
- Use Pearson correlation, ρ, as test statistic
- Compute p-value as fraction of replicates that have ρ at least as large as observed.

More populous counties voted for Obama



More populous counties voted for Obama



I did this procedure, and in all 10,000 of my replicates under the null hypothesis, not one had a Pearson correlation coefficient as high as the observed value of 0.54.

I tried generating 100,000, and then a million replicates. In all cases, not one replicate had a Pearson correlation coefficient as high as point-54.

This does not mean that the p-value is zero. It means that it is so low that we would have to generate an enormous number of replicates to have even one that has a test statistic sufficiently extreme

p-value is very very small

Let's practice!

STATISTICAL THINKING IN PYTHON (PART 2)