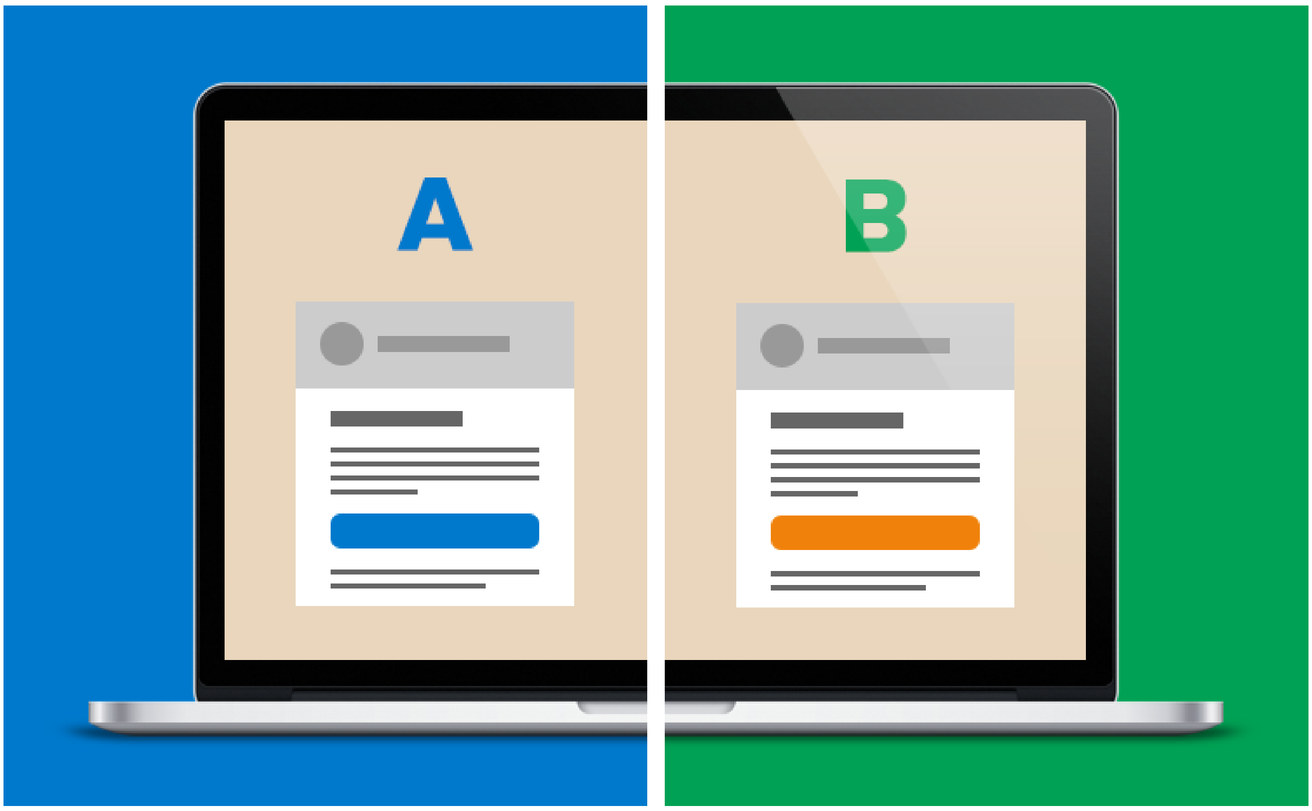
Handy Functions for A/B Testing in Python

Henry Feng
Mar 4, 2019 · 5 min read

```
Feel free to follow my Medium to get instant notification :)
```

Recently, I took the "Customer Analytics & A/B Testing in Python" course on DataCamp, trying to equip myself with more ideas and skills regarding A/B testing and experimentation. In the course, the instructor provides several functions to calculate important statistics. I just copied those functions in my Spyder, ran several examples to test their usability and share those functions here to benefit those who are interested in A/B testing.



Power function

The power of a hypothesis test is the probability of making the correct decision if the alternative hypothesis is true. In general, for every hypothesis test we conduct, we will try to maximize the power. Typically, we desire the power to be 0.8 or greater.

The get_power function will return the power given the argument sample size(n), control group mean (p1), test_group mean (p2), and confidence interval (cl).

```
from scipy import stats

def get_power(n, p1, p2, cl):
    alpha = 1 - cl
    qu = stats.norm.ppf(1 - alpha/2)
    diff = abs(p2-p1)
    bp = (p1+p2) / 2

    v1 = p1 * (1-p1)
    v2 = p2 * (1-p2)
    bv = bp * (1-bp)

    power_part_one = stats.norm.cdf((n**0.5 * diff - qu * (2 *
bv)**0.5) / (v1+v2) ** 0.5)
    power_part_two = 1 - stats.norm.cdf((n**0.5 * diff + qu * (2 *
bv)**0.5) / (v1+v2) ** 0.5)

    power = power_part_one + power_part_two

    return (power)
```

I just passed into different arguments and see what the power the function returns.

```
get_power(1000, 0.1, 0.12, 0.95)   #  0.29808032538146
get_power(2000, 0.1, 0.12, 0.95)   #  0.524515256115834
get_power(1000, 0.1, 0.12, 0.8)    #  0.5621010118690234
```

From the results, I conclude two points. One is that the bigger the sample size is, the bigger the power. Two is that the bigger the confidence level, the smaller the power.

It is known that one minus confidence level is the type one error. If a confidence level is smaller, which indicates the larger type one error (Alpha). The larger the type one error, the smaller the type two error (Beta). Power is calculated as one minus Beta. Therefore, the smaller the Beta, the bigger the power.

Sample size function

When conducting an experiment, it is very important to think about sample size. And the function provided by the course is able to create the proper sample size based on power, control group mean, test group mean and confidence level.

The sample size the function returns indicates that under the desired power, if analysts want to observe the change in the mean between test groups and control groups for certain confidence interval, the analyst will need N sample.

```
def get_sample_size(power, p1, p2, cl, max_n=1000000):
    n = 1
    while n <= max_n:
        tmp_power = get_power(n, p1, p2, cl)

if tmp_power >= power:
            return n
        else:
            n = n + 100

return "Increase Max N Value"
```

Let’s try some different numbers into this function.

```
# Trial1
conversion_rate = 0.03
power = 0.8
cl = 0.9
percent_lift = 0.1
conversion_rate_p2 = conversion_rate * (1 + percent_lift)

get_sample_size(power, conversion_rate, conversion_rate_p2, cl)
# =>>>>>>> 42001

# Trial2
conversion_rate = 0.03
power = 0.95
cl = 0.9
percent_lift = 0.1
conversion_rate_p2 = conversion_rate * (1 + percent_lift)

get_sample_size(power, conversion_rate, conversion_rate_p2, cl)
# =>>>>>> 73401
```

The result is not surprising. If a product analyst desires a higher power, bigger sample sizes are needed.

P-value Function

P-value, in a technical term, is the probability of obtaining an effect at least as extreme as the one in your sample data, assuming the truth of the null hypothesis. P-value is widely used in different statistical tests for showing significance.

The p-value function here is to test the result value between test groups and control groups, given the sample size of each group, whether the change is significant enough to reach the A/B testing conclusion.

```
def get_pvalue(con_conv, test_conv, con_size, test_size):
    lift = -abs(test_conv - con_conv)

    scale_one = con_conv * (1-con_conv) * (1/ con_size)
    scale_two = test_conv * (1-test_conv) * (1/ test_size)
    scale_val = (scale_one + scale_two) ** 0.5

    p_value = 2 * stats.norm.cdf(lift, loc=0, scale = scale_val)
    return p_value
```

Let's try some some numbers. With the same conversion rate of the test group and control group. I try different sample sizes. The result quite matched I've expected.

```
# Trial 1
con_conv = 0.034351
test_conv = 0.041984
con_size = 48236
test_size = 49867

get_pvalue(con_conv, test_conv, con_size, test_size)
# 4.257297485586909e-10

# Trial 2
con_conv = 0.034351
test_conv = 0.041984
con_size = 48
test_size = 49

get_pvalue(con_conv, test_conv, con_size, test_size)
# 0.8443
```

With the smaller sample sizes, it is observed that the result will turn into not significant compared to the bigger sample size. Therefore, if a product analyst wants to detect the significance in changes among test and control groups, the larger the sample the better.

Confidence Interval Function

The last handy function for A/B testing setting is the confidence interval function. Confidence interval will grant the analyst a range of estimation, and it somehow reveals the snapshot of the population mean. Below is the code provided by DataCamp.

```
def get_ci(lift, alpha, sd):
    val = abs(stats.norm.ppf((1-alpha)/2))

    lwr_bnd = lift - val * sd
    upr_bnd = lift + val * sd

    return (lwr_bnd, upr_bnd)
```

Finally, let’s try out this function.

```
# Trial 1
test_conv = 0.102005
con_conv = 0.090965
test_size = 56350
con_size = 58583

lift_mean = test_conv - con_conv
lift_variance = (1 - test_conv) * test_conv /test_size + (1 -
con_conv) * con_conv / con_size
lift_sd = lift_variance**0.5

get_ci(lift_mean, 0.95, lift_sd)

## (0.007624337671217316, 0.014455662328782672)

# Trial 2
test_conv = 0.102005
con_conv = 0.090965
test_size = 563
con_size = 585

lift_mean = test_conv - con_conv
lift_variance = (1 - test_conv) * test_conv /test_size + (1 -
con_conv) * con_conv / con_size
lift_sd = lift_variance**0.5
```

```
get_ci(lift_mean, 0.95, lift_sd)

## (-0.023135997406420666, 0.045215997406420655)
```

As you can see, if the sample size is tuned into smaller ones, the interval the function returns will be bigger. With the bigger sample size, the interval will be tighter, which means the sample sizes are closer to the population size, and it doesn't need a wider interval to capture the true mean.

Conclusion

In this brief article, I documented what I've learned from DataCamp course. This function might be tiny and simple, but I believe they really lay the foundation of a good experiment. Therefore, feel free to utilize these sources and examine your A/B testing setting and results. Happy analyzing.

yunhanfeng/ab_testing_function

Extract the code from DataCamp course, some handy functions for A/B testing statistics are shown here ...

github.com

A B Testing Experiment Product Analytics Data Science Python