UDACITY

Return to Classroom

DISCUSS ON STUDENT HUB

Analyze A/B Test Results

REVIEW

HISTORY

Meets Specifications

Congratulations, you have done a great job with this project. Your exploration of the data was interesting, as well as insightful. It is clear that you have put a great amount of effort and thought into this project. You proved deep comprehension with A/B testing by following the right approach to get the right answer for the analysis questions.

Check tutorials for implementing A/B testing with Python:

- Implementing A/B Tests in Python
- The Math Behind A/B Testing with Example Python Code
- How to perform an A/B test correctly in Python

Keep up the Hard Work

Code Quality

All code cells can be run without error.

All code cells run without errors and produce the required results.

Docstrings, comments, and variable names enable readability of the code.

All the code cells are documented properly, it's highly recommended and one of the coding best practices to document your work along the process using comments & docstrings to justify the code functionality. Clarification comments are intended for anyone (including your future self) who may need to maintain, refactor, or extend your code, check this article for more information about the best approaches to document your code: **How to document source code responsibly**

Statistical Analyses

All results from different analyses are correctly interpreted.

Great work, all the results of the statistical analyses have good interpretation with clear and concise conclusions.

Suggestion(Optional)

A faster way to simulate the 10000 trials

- When possible, it is always more computationally efficient to use NumPy built-in operations over explicit `for` loops. The short reason is that NumPy -based operations attack a computational problem based on vectors by computing large chunks simultaneously.
- Additionally, using loops to simulate 10000 can take a considerable amount of time vs using NumPy
https://softwareengineering.stackexchange.com/questions/254475/how-do-i-move-away-from-the-for-loop-school-of-thought

```
new_converted_simulation = np.random.binomial(n_new, p_new, 10000)/n_new
old_converted_simulation = np.random.binomial(n_old, p_old, 10000)/n_old
p_diffs = new_converted_simulation - old_converted_simulation
```

- Essentially, we are applying the null proportion to the total size of each page using the binomial distribution. Each element, for example, in `np.random.binomial(n_new, p_new, 10000)` results in an array with values like `[17262, 17250, 17277...]`. This array is 10000 elements large
- When we divide it by `n_new`, Python broadcasts `n_new` for each element and we return a proportion for each element.
- This is essentially simulating, 10000, the new page conversion rate.
- We do this again for the old page.
- The difference between the two will result in a simulated difference array of length 10000 between the new page and old page conversions.
- Note that this method does not require you to calculate the null values to get the p-value.

For all numeric values, you should provide the correct results of the analysis.

All calculations are correctly reached especially that for the simulation, and z-test, check an interesting reading about hypothesis testing for beginners: **All about hypothesis testing, from p-values to Z-test** all-in-one

Conclusions should include not only statistical reasoning, but also practical reasoning for the situation.

- Spot On!!! Great intuition with the relationship between the different hypotheses statements.
- Extra Credit Knowing that Part iii is a two-tailed test and Part ii is a one-tail test, can you convert the p-values between each other? check an interesting paper discussing this method: **One-Tailed and Two-Tailed Results**

DOWNLOAD PROJECT

RETURN TO PATH

Rate this review

START