

Analyze A/B Test Results

This project will assure you have mastered the subjects covered in the statistics lessons. We have organized the current notebook into the following sections:

- [Introduction](#)
- [Part I - Probability](#)
- [Part II - A/B Test](#)
- [Part III - Regression](#)
- [Final Check](#)
- [Submission](#)

Specific programming tasks are marked with a **ToDo** tag.

Introduction

A/B tests are very commonly performed by data analysts and data scientists. For this project, you will be working to understand the results of an A/B test run by an e-commerce website. Your goal is to work through this notebook to help the company understand if they should:

- Implement the new webpage,
- Keep the old webpage, or
- Perhaps run the experiment longer to make their decision.

Part I - Probability

To get started, let's import our libraries.

```
In [1]: import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt

%matplotlib inline
#We are setting the seed to assure you get the same answers on quizzes as we set up
random.seed(42)
```

ToDo 1.1

a. Read in the dataset from the `ab_data.csv` file and take a look at the top few rows here:

```
In [2]: df = pd.read_csv('ab_data.csv')
df.head()
```

```
Out[2]:
```

	user_id	timestamp	group	landing_page	converted
0	851104	2017-01-21 22:11:48.556739	control	old_page	0
1	804228	2017-01-12 08:01:45.159739	control	old_page	0
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0

3	853541	2017-01-08 18:28:03.143765	treatment	new_page	0
4	864975	2017-01-21 01:52:26.210827	control	old_page	1

b. Use the cell below to find the number of rows in the dataset.

```
In [3]: df.shape[0]
```

```
Out[3]: 294478
```

c. The number of unique users in the dataset.

```
In [4]: df.user_id.nunique()
```

```
Out[4]: 290584
```

d. The proportion of users converted.

```
In [5]: df.query('converted == 1').shape[0]/df.shape[0]
```

```
Out[5]: 0.11965919355605512
```

e. The number of times when the "group" is `treatment` but "landing_page" is not a `new_page`.

```
In [6]: df.query('group == "treatment" & landing_page != "new_page").shape[0]
```

```
Out[6]: 1965
```

The number of times when the "group" is `control` but "landing_page" is not an `old_page`.

```
In [7]: df.query('group == "control" & landing_page != "old_page").shape[0]
```

```
Out[7]: 1928
```

f. Do any of the rows have missing values?

```
In [8]: df.isna().sum()
```

```
Out[8]: user_id      0
timestamp  0
group      0
landing_page  0
converted  0
dtype: int64
```

ToDo 1.2

In a particular row, the **group** and **landing_page** columns should have either of the following acceptable values:

user_id	timestamp	group	landing_page	converted
XXXX	XXXX	control	old_page	X
XXXX	XXXX	treatment	new_page	X

It means, the `control` group users should match with `old_page`; and `treatment` group users should matched with the `new_page`.

However, for the rows where `treatment` does not match with `new_page` or `control` does not match with `old_page`, we cannot be sure if such rows truly received the new or old webpage.

a. Now use the answer to the quiz to create a new dataset that meets the specifications from the quiz. Store your new dataframe in **df2**.

```
In [9]: # Remove the inaccurate rows, and store the result in a new dataframe df2
mismatch1 = df.query('landing_page == "new_page" & group != "treatment"')
mismatch2 = df.query('landing_page == "old_page" & group != "control"')
mismatch2.head()
```

```
Out[9]:
```

	user_id	timestamp	group	landing_page	converted
308	857184	2017-01-20 07:34:59.832626	treatment	old_page	0
327	686623	2017-01-09 14:26:40.734775	treatment	old_page	0
357	856078	2017-01-12 12:29:30.354835	treatment	old_page	0
685	666385	2017-01-23 08:11:54.823806	treatment	old_page	0
713	748761	2017-01-10 15:47:44.445196	treatment	old_page	0

```
In [10]: mismatch1.shape[0], mismatch2.shape[0]
```

```
Out[10]: (1928, 1965)
```

```
In [11]: df = df.drop(mismatch1.index)
df = df.drop(mismatch2.index)
df2 = df
```

```
In [12]: # Double Check all of the incorrect rows were removed from df2 -
# Output of the statement below should be 0
df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].sha
```

```
Out[12]: 0
```

ToDo 1.3

a. How many unique **user_ids** are in **df2**?

```
In [13]: df2.user_id.nunique()
```

```
Out[13]: 290584
```

b. There is one **user_id** repeated in **df2**. What is it?

```
In [14]: df.user_id.duplicated().sum()
```

```
Out[14]: 1
```

```
In [15]: df[df.user_id.duplicated()]
```

```
Out[15]:
```

	user_id	timestamp	group	landing_page	converted
--	---------	-----------	-------	--------------	-----------

c. Display the rows for the duplicate **user_id**?

```
In [16]: df[df.user_id.duplicated(keep=False)]
```

```
Out[16]:
```

	user_id	timestamp	group	landing_page	converted
1899	773192	2017-01-09 05:37:58.781806	treatment	new_page	0
2893	773192	2017-01-14 02:55:59.590927	treatment	new_page	0

d. Remove **one** of the rows with a duplicate **user_id**, from the **df2** dataframe.

```
In [17]: # Remove one of the rows with a duplicate user_id..
df2.drop_duplicates(subset='user_id', inplace=True)
# Check again if the row with a duplicate user_id is deleted or not
df.user_id.duplicated().sum()
```

```
Out[17]: 0
```

ToDo 1.4

a. What is the probability of an individual converting regardless of the page they receive?

```
In [18]: df2.query('converted == 1').shape[0]/df2.shape[0]
```

```
Out[18]: 0.11959708724499628
```

b. Given that an individual was in the **control** group, what is the probability they converted?

```
In [19]: df2.query('converted == 1 & group == "control").shape[0]/df2.query('group == "control"').shape[0]
```

```
Out[19]: 0.1203863045004612
```

c. Given that an individual was in the **treatment** group, what is the probability they converted?

```
In [20]: df2.query('converted == 1 & group == "treatment").shape[0]/df2.query('group == "treatment"').shape[0]
```

```
Out[20]: 0.11880806551510564
```

Calculate the actual difference (**obs_diff**) between the conversion rates for the two groups. You will need that later.

```
In [21]: # Calculate the actual difference (obs_diff) between the conversion rates for the two groups
obs_diff = df2.query('converted == 1 & group == "treatment").shape[0]/df2.query('group == "treatment"').shape[0] -
df2.query('converted == 1 & group == "control").shape[0]/df2.query('group == "control"').shape[0]
```

```
Out[21]: -0.0015782389853555567
```

d. What is the probability that an individual received the new page?

```
In [22]: df2.query('landing_page == "new_page").shape[0]/df2.shape[0]
```

```
Out[22]: 0.5000619442226688
```

e. Consider your results from parts (a) through (d) above, and explain below whether the new treatment group users lead to more conversions.

From what we can see above, there is no sufficient evidence to confirm that the new page leads to more conversions.

Part II - A/B Test

Since a timestamp is associated with each event, you could run a hypothesis test continuously as long as you observe the events.

However, then the hard questions would be:

- Do you stop as soon as one page is considered significantly better than another or does it need to happen consistently for a certain amount of time?
- How long do you run to render a decision that neither page is better than another?

These questions are the difficult parts associated with A/B tests in general.

ToDo 2.1

For now, consider you need to make the decision just based on all the data provided.

Recall that you just calculated that the "converted" probability (or rate) for the old page is *slightly* higher than that of the new page (ToDo 1.4.c).

If you want to assume that the old page is better unless the new page proves to be definitely better at a Type I error rate of 5%, what should be your null and alternative hypotheses (H_0 and H_1)?

You can state your hypothesis in terms of words or in terms of p_{old} and p_{new} , which are the "converted" probability (or rate) for the old and new pages respectively.

ToDo 2.2 - Null Hypothesis H_0 Testing

Under the null hypothesis H_0 , assume that p_{new} and p_{old} are equal. Furthermore, assume that p_{new} and p_{old} both are equal to the **converted** success rate in the `df2` data regardless of the page. So, our assumption is:

$$p_{new} = p_{old} = p_{population}$$

In this section, you will:

- Simulate (bootstrap) sample data set for both groups, and compute the "converted" probability p for those samples.
- Use a sample size for each group equal to the ones in the `df2` data.
- Compute the difference in the "converted" probability for the two samples above.

- Perform the sampling distribution for the "difference in the converted probability" between the two simulated-samples over 10,000 iterations; and calculate an estimate.

Use the cells below to provide the necessary parts of this simulation. You can use **Quiz 5** in the classroom to make sure you are on the right track.

$$p_{new} = p_{old} = p_{population}$$

a. What is the **conversion rate for p_{new} under the null hypothesis?**

```
In [23]: p_new = df2.query('converted == 1').shape[0]/df2.shape[0]
p_new
```

```
Out[23]: 0.11959708724499628
```

b. What is the **conversion rate for p_{old} under the null hypothesis?**

```
In [24]: p_old = df2.query('converted == 1').shape[0]/df2.shape[0]
p_old
```

```
Out[24]: 0.11959708724499628
```

c. What is n_{new} , the number of individuals in the treatment group?

```
In [25]: n_new = df2.query('group == "treatment"').shape[0]
n_new
```

```
Out[25]: 145310
```

d. What is n_{old} , the number of individuals in the control group?

```
In [26]: n_old = df2.query('group == "control"').shape[0]
n_old
```

```
Out[26]: 145274
```

e. Simulate Sample for the **treatment Group**

Simulate n_{new} transactions with a conversion rate of p_{new} under the null hypothesis.

Store these n_{new} 1's and 0's in the **new_page_converted** numpy array.

```
In [27]: # Simulate a Sample for the treatment Group
new_page_converted = np.random.choice(2, n_new, p=(1-p_new, p_new))
new_page_converted
```

```
Out[27]: array([0, 0, 0, ..., 0, 0, 0])
```

f. Simulate Sample for the **control Group**

Simulate n_{old} transactions with a conversion rate of p_{old} under the null hypothesis.

Store these n_{old} 1's and 0's in the **old_page_converted** numpy array.

```
In [28]: # Simulate a Sample for the control Group
old_page_converted = np.random.choice(2, n_old, p=(1-p_old, p_old))
```

g. Find the difference in the "converted" probability ($p'_{new} - p'_{old}$) for your simulated samples from the parts (e) and (f) above.

```
In [29]: new_page_converted.mean() - old_page_converted.mean()
Out[29]: 0.00013536275785740182
```

h. Sampling distribution

Re-create `new_page_converted` and `old_page_converted` and find the ($p'_{new} - p'_{old}$) value 10,000 times using the same simulation process you used in parts (a) through (g) above.

Store all ($p'_{new} - p'_{old}$) values in a NumPy array called `p_diffs`.

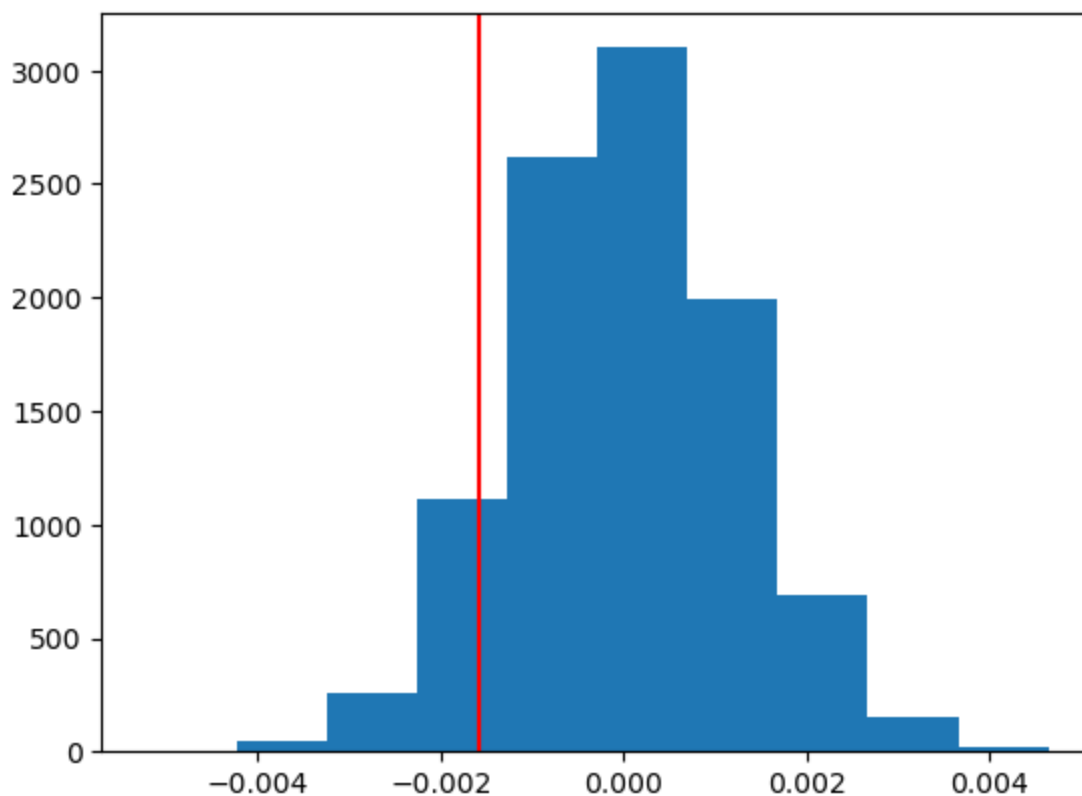
```
In [30]: # Sampling distribution
p_diffs = []
new_page_converted = np.random.binomial(n_new, p_new, 10000)/n_new
old_page_converted = np.random.binomial(n_old, p_old, 10000)/n_old
p_diffs = new_page_converted - old_page_converted
```

i. Histogram

Plot a histogram of the `p_diffs`. Does this plot look like what you expected? Use the matching problem in the classroom to assure you fully understand what was computed here.

Also, use `plt.axvline()` method to mark the actual difference observed in the `df2` data (recall `obs_diff`), in the chart.

```
In [31]: p_diffs = np.array(p_diffs)
plt.hist(p_diffs)
plt.axvline(x=obs_diff, color="r");
```



j. What proportion of the `p_diffs` are greater than the actual difference observed in the `df2` data?

```
In [32]: (p_diffs > obs_diff).mean()
```

```
Out[32]: 0.9071
```

k. Please explain in words what you have just computed in part **j** above.

- What is this value called in scientific studies?
- What does this value signify in terms of whether or not there is a difference between the new and old pages?

Above value is called pvalue which is what we calculate given that the Null (H_0) hypothesis is true. This value here signifies that with a type I error, we fail to reject the Null (H_0) hypothesis as on average, the old page has a higher conversion rate than the new page.

I. Using Built-in Methods for Hypothesis Testing

We could also use a built-in to achieve similar results. Though using the built-in might be easier to code, the above portions are a walkthrough of the ideas that are critical to correctly thinking about statistical significance.

Fill in the statements below to calculate the:

- `convert_old` : number of conversions with the old_page
- `convert_new` : number of conversions with the new_page
- `n_old` : number of individuals who were shown the old_page
- `n_new` : number of individuals who were shown the new_page

```
In [33]: # number of conversions with the old_page
convert_old = df2.query('landing_page == "old_page"')['converted'].sum()

# number of conversions with the new_page
convert_new = df2.query('landing_page == "new_page"')['converted'].sum()

# number of individuals who were shown the old_page
n_old = df2.query('landing_page == "old_page"').shape[0]

# number of individuals who received new_page
n_new = df2.query('landing_page == "new_page"').shape[0]
```

m. Now use `sm.stats.proportions_ztest()` to compute your test statistic and p-value. [Here](#) is a helpful link on using the built in.

The syntax is:

```
proportions_ztest(count_array, nobs_array, alternative='larger')
```

where,

- `count_array` = represents the number of "converted" for each group
- `nobs_array` = represents the total number of observations (rows) in each group
- `alternative` = choose one of the values from `['two-sided', 'smaller', 'larger']` depending upon two-tailed, left-tailed, or right-tailed respectively.

The built-in function above will return the `z_score`, `p_value`.

About the two-sample z-test

Recall that you have plotted a distribution `p_diffs` representing the difference in the "converted" probability ($p'_{new} - p'_{old}$) for your two simulated samples 10,000 times.

Another way for comparing the mean of two independent and normal distribution is a **two-sample z-test**. You can perform the Z-test to calculate the Z_score, as shown in the equation below:

$$Z_{score} = \frac{(p'_{new} - p'_{old}) - (p_{new} - p_{old})}{\sqrt{\frac{\sigma_{new}^2}{n_{new}} + \frac{\sigma_{old}^2}{n_{old}}}}$$

where,

- p' is the "converted" success rate in the sample
- p_{new} and p_{old} are the "converted" success rate for the two groups in the population.
- σ_{new} and σ_{old} are the standard deviation for the two groups in the population.
- n_{new} and n_{old} represent the size of the two groups or samples (it's same in our case)

Z-test is performed when the sample size is large, and the population variance is known. The z-score represents the distance between the two "converted" success rates in terms of the standard error.

Next step is to make a decision to reject or fail to reject the null hypothesis based on comparing these two values:

- Z_{score}
- Z_{α} or $Z_{0.05}$, also known as critical value at 95% confidence interval. $Z_{0.05}$ is 1.645 for one-tailed tests, and 1.960 for two-tailed test. You can determine the Z_{α} from the z-table manually.

Decide if your hypothesis is either a two-tailed, left-tailed, or right-tailed test. Accordingly, reject OR fail to reject the null based on the comparison between Z_{score} and Z_{α} . We determine whether or not the Z_{score} lies in the "rejection region" in the distribution. In other words, a "rejection region" is an interval where the null hypothesis is rejected if the Z_{score} lies in that region.

Reference:

- Example 9.1.2 on this [page/09%3A_Two-Sample_Problems/9.01%3A_Comparison_of_Two_Population_Means-_Large_Independent_Samples](https://www.stats.libretexts.org/Bookshelves/Introductory_Statistics/Book%3A_Two-Sample_Problems/9.01%3A_Comparison_of_Two_Population_Means-_Large_Independent_Samples)), courtesy www.stats.libretexts.org

```
In [34]: import statsmodels.api as sm;
# ToDo: Complete the sm.stats.proportions_ztest() method arguments
z_score, p_value = sm.stats.proportions_ztest([convert_new, convert_old], [n_new, n_old])
print(z_score, p_value)

-1.3109241984234394 0.9050583127590245
```

n. What do the z-score and p-value you computed in the previous question mean for the conversion rates of the old and new pages? Do they agree with the findings in parts **j.** and **k.**?

A z-score describes the position of a raw score in terms of its distance from the mean, when measured in standard deviation units. Here, the difference between our statistic and null hypothesis is -1.311 standard

deviation units above the mean which is less than Z_α that has a critical value of 1.645 for a one-sided test. Also, our pvalue (0.91) is higher than α for a type I error. Therefore, we fail to reject the null hypothesis.

Part III - A regression approach

ToDo 3.1

a. Since each row in the `df2` data is either a conversion or no conversion, what type of regression should you be performing in this case?

We'll be using logestic regression as we'll be comparing categorical values.

b. The goal is to use **statsmodels** library to fit the regression model you specified in part **a.** above to see if there is a significant difference in conversion based on the page-type a customer receives. However, you first need to create the following two columns in the `df2` dataframe:

1. `intercept` - It should be `1` in the entire column.
2. `ab_page` - It's a dummy variable column, having a value `1` when an individual receives the **treatment**, otherwise `0`.

```
In [35]: df2['intercept'] = 1
df2[['a_page', 'ab_page']] = pd.get_dummies(df2['group'])
df2.drop('a_page', axis=1, inplace=True)
```

```
In [36]: df2.head()
```

```
Out[36]:
```

	user_id	timestamp	group	landing_page	converted	intercept	ab_page
0	851104	2017-01-21 22:11:48.556739	control	old_page	0	1	0
1	804228	2017-01-12 08:01:45.159739	control	old_page	0	1	0
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0	1	1
3	853541	2017-01-08 18:28:03.143765	treatment	new_page	0	1	1
4	864975	2017-01-21 01:52:26.210827	control	old_page	1	1	0

c. Use **statsmodels** to instantiate your regression model on the two columns you created in part (b). above, then fit the model to predict whether or not an individual converts.

```
In [37]: logit_mod = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])
results = logit_mod.fit()
```

```
Optimization terminated successfully.
      Current function value: 0.366118
      Iterations 6
```

d. Provide the summary of your model below, and use it as necessary to answer the following questions.

```
In [38]: results.summary2()
```

```
Out[38]:
```

Model:	Logit	Pseudo R-squared:	0.000
Dependent Variable:	converted	AIC:	212780.3502
Date:	2023-07-09 12:42	BIC:	212801.5095

No. Observations:	290584	Log-Likelihood:	-1.0639e+05
Df Model:	1	LL-Null:	-1.0639e+05
Df Residuals:	290582	LLR p-value:	0.18988
Converged:	1.0000	Scale:	1.0000
No. Iterations:	6.0000		

	Coef.	Std.Err.	z	P> z	[0.025	0.975]
intercept	-1.9888	0.0081	-246.6690	0.0000	-2.0046	-1.9730
ab_page	-0.0150	0.0114	-1.3109	0.1899	-0.0374	0.0074

e. What is the p-value associated with **ab_page**? Why does it differ from the value you found in **Part II**?

The pvalue associated with ab_page is 0.1899, and here it differs our previous pvalue of 0.905 in Part II as here it was a two-sided test, while in Part II it was a one-sided test.

Our null hypothesis here in Part III is:

$$H_0: P_{new} - P_{old} = 0$$

$$H_1: P_{new} - P_{old} \neq 0$$

However, in Part II it was:

$$H_0: P_{new} - P_{old} \leq 0$$

$$H_1: P_{new} - P_{old} > 0$$

f. Now, you are considering other things that might influence whether or not an individual converts. Discuss why it is a good idea to consider other factors to add into your regression model. Are there any disadvantages to adding additional terms into your regression model?

There could be other factors that affects when would an individual converts such as their age, gender, and location. The disadvantage in that is that there could be correlation between those terms, which would cause collinearity.

g. Adding countries

Now along with testing if the conversion rate changes for different pages, also add an effect based on which country a user lives in.

1. You will need to read in the **countries.csv** dataset and merge together your **df2** datasets on the appropriate rows. You call the resulting dataframe **df_merged**. [Here](#) are the docs for joining tables.
2. Does it appear that country had an impact on conversion? To answer this question, consider the three unique values, **['UK', 'US', 'CA']**, in the **country** column. Create dummy variables for these country columns.

Provide the statistical output as well as a written response to answer this question.

```
In [39]: # Read the countries.csv
countries = pd.read_csv('countries.csv')
```

```
countries.head()
```

```
Out[39]:
```

	user_id	country
0	834778	UK
1	928468	US
2	822059	UK
3	711597	UK
4	710616	UK

```
In [40]: # Join with the df2 dataframe
df_merged = df2.set_index('user_id').join(countries.set_index('user_id'))
```

```
In [41]: # Create the necessary dummy variables
df_merged[['CA', 'UK', 'US']] = pd.get_dummies(df_merged['country'])
```

```
In [42]: # Fit your model, and summarize the results
logit_mod = sm.Logit(df_merged['converted'], df_merged[['intercept', 'ab_page', 'CA', 'U
results = logit_mod.fit()
results.summary2()
```

Optimization terminated successfully.
Current function value: 0.366113
Iterations 6

```
Out[42]:
```

Model:	Logit	Pseudo R-squared:	0.000
--------	-------	-------------------	-------

Dependent Variable:	converted	AIC:	212781.1253
---------------------	-----------	------	-------------

Date:	2023-07-09 12:42	BIC:	212823.4439
-------	------------------	------	-------------

No. Observations:	290584	Log-Likelihood:	-1.0639e+05
-------------------	--------	-----------------	-------------

Df Model:	3	LL-Null:	-1.0639e+05
-----------	---	----------	-------------

Df Residuals:	290580	LLR p-value:	0.17599
---------------	--------	--------------	---------

Converged:	1.0000	Scale:	1.0000
------------	--------	--------	--------

No. Iterations:	6.0000
-----------------	--------

	Coef.	Std.Err.	z	P> z	[0.025	0.975]
intercept	-1.9893	0.0089	-223.7628	0.0000	-2.0067	-1.9718
ab_page	-0.0149	0.0114	-1.3069	0.1912	-0.0374	0.0075
CA	-0.0408	0.0269	-1.5161	0.1295	-0.0934	0.0119
UK	0.0099	0.0133	0.7433	0.4573	-0.0162	0.0359

From the pvalues above, it doesn't seem that countries had an impact on conversion as all pvalues are larger than our $\alpha_{0.05}$.

h. Fit your model and obtain the results

Though you have now looked at the individual factors of country and page on conversion, we would now like to look at an interaction between page and country to see if are there significant effects on conversion.

Create the necessary additional columns, and fit the new model.

Provide the summary results (statistical output), and your conclusions (written response) based on the results.

```
In [43]: df_merged['US_page'] = df_merged['US'] * df_merged['ab_page']
df_merged['UK_page'] = df_merged['UK'] * df_merged['ab_page']
df_merged['CA_page'] = df_merged['CA'] * df_merged['ab_page']
```

```
In [44]: # Fit your model, and summarize the results
logit_mod = sm.Logit(df_merged['converted'], df_merged[['intercept', 'CA_page', 'UK_page']])
results = logit_mod.fit()
results.summary2()
```

Optimization terminated successfully.
Current function value: 0.366113
Iterations 6

```
Out[44]:
```

Model:	Logit	Pseudo R-squared:	0.000
--------	-------	-------------------	-------

Dependent Variable:	converted	AIC:	212779.0384
---------------------	-----------	------	-------------

Date:	2023-07-09 12:42	BIC:	212810.7773
-------	------------------	------	-------------

No. Observations:	290584	Log-Likelihood:	-1.0639e+05
-------------------	--------	-----------------	-------------

Df Model:	2	LL-Null:	-1.0639e+05
-----------	---	----------	-------------

Df Residuals:	290581	LLR p-value:	0.080850
---------------	--------	--------------	----------

Converged:	1.0000	Scale:	1.0000
------------	--------	--------	--------

No. Iterations:	6.0000
-----------------	--------

	Coef.	Std.Err.	z	P> z	[0.025	0.975]
intercept	-1.9963	0.0062	-322.0487	0.0000	-2.0084	-1.9841
CA_page	-0.0752	0.0376	-1.9974	0.0458	-0.1489	-0.0014
UK_page	0.0149	0.0173	0.8617	0.3888	-0.0190	0.0488

```
In [45]: 1/np.exp(results.params)
```

```
Out[45]:
```

intercept	7.361591
CA_page	1.078076
UK_page	0.985222
dtype:	float64

Project Summary

After adding an interaction between page, and country we can see a statistical significance in number of individuals converting when they receive the new page when their their location is Canada as it's 1.08 times more conversions compared to those who are in USA with a pvalue of 0.046 which is less than our $\alpha_{0.05}$. However, this is only for Canada, while it may seem as a statistical significant on it's own, but for all users who visit the page from USA and UK, there is no statistical significant as US_page and UKpage pvalues are larger than $\alpha_{0.05}$.

Based on what we've computed, we fail to reject the null hypothesis, even though it was very useful to predict conversions based on country, for all users, it's not practically significant, and therefor we shouldn't switch to using the new page, when the old one performs better overall.