Analyze_ab_test_results_notebook

September 22, 2022

1 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:

- Section **??**
- Section **??**
- Section **??**
- Section **??**
- Section **??**
- Section **??**
- Section **??**

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.

Each **ToDo** task below has an associated quiz present in the classroom. Though the classroom quizzes are **not necessary** to complete the project, they help ensure you are on the right track as you work through the project, and you can feel more confident in your final submission meeting the rubric specification.

Part I - Probability

To get started, let's import our libraries.

```
In [86]: 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)
```

1.0.1 ToDo 1.1

Now, read in the `ab_data.csv` data. Store it in `df`. Below is the description of the data, there are a total of 5 columns:

Data columns	Purpose	Valid values
user_id	Unique ID	Int64 values
timestamp	Time stamp when the user visited the webpage	-
group	In the current A/B experiment, the users are categorized into two broad groups. The `control` group users are expected to be served with `old_page`; and `treatment` group users are matched with the `new_page`. However, **some inaccurate rows** are present in the initial data, such as a `control` group user is matched with a `new_page`.	`['control', 'treatment']`
landing_page	It denotes whether the user visited the old or new webpage.	`['old_page', 'new_page']`
converted	It denotes whether the user decided to pay for the company's product. Here, `1` means yes, the user bought the product.	`[0, 1]`

Use your dataframe to answer the questions in Quiz 1 of the classroom.

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

```
In [87]: # import our dataset
         df = pd.read_csv('ab_data.csv')
```

```
df.head()
```

```
Out[87]:    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 [88]: # number of rows in the dataset is 294478
         df.shape[0]

Out[88]: 294478
```

c. The number of unique users in the dataset.

```
In [89]: # number of unique users are 290584
         df.user_id.value_counts().shape[0]

Out[89]: 290584
```

d. The proportion of users converted.

```
In [90]: # calculated number of converted users (0.119659)
         df.converted.value_counts(normalize = True)

Out[90]: 0    0.880341
         1    0.119659
         Name: converted, dtype: float64
```

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

```
In [91]: # number of times where treatment group does not match with the new page
         df.query("group == 'treatment' and landing_page == 'old_page'").shape[0]

Out[91]: 1965
```

f. Do any of the rows have missing values?

```
In [92]: # view if there are any missing values
         df.isna().sum()

Out[92]: user_id         0
         timestamp       0
         group           0
         landing_page    0
         converted       0
         dtype: int64
```

1.0.2 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 wepage.

Use **Quiz 2** in the classroom to figure out how should we handle the rows where the group and landing_page columns don't match?

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 [93]: # Remove the inaccurate rows, and store the result in a new dataframe df2

         df_1 = df.drop(df[(df.group == 'control') & (df.landing_page == 'new_page')].index)

         df2 = df_1.drop(df_1[(df.group == 'treatment') & (df_1.landing_page == 'old_page')].ind

/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:5: UserWarning: Boolean Series key
  """
```

```
In [94]: # 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].sh

Out[94]: 0
```

1.0.3 ToDo 1.3

Use **df2** and the cells below to answer questions for **Quiz 3** in the classroom.

a. How many unique **user_id**s are in **df2**?

```
In [95]: # Get the number of unique user in df2
         df2.user_id.value_counts().shape[0]

Out[95]: 290584
```

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

```
In [96]: # Find the duplicated user_id
         df2[df2.duplicated(['user_id'])].user_id

Out[96]: 2893    773192
         Name: user_id, dtype: int64
```

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

In [97]:
```
# View rows of duplicated user
df2[df2.duplicated(['user_id'], keep = False)]
```

Out[97]:

	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 [98]:
```
# Remove one of the rows with a duplicate user_id..
# Hint: The dataframe.drop_duplicates() may not work in this case because the rows with
df2.drop(df2[df2.duplicated(['user_id'])].index, inplace = True)
# Check again if the row with a duplicate user_id is deleted or not
df2[df2.duplicated(['user_id'])]
```

Out[98]:
```
Empty DataFrame
Columns: [user_id, timestamp, group, landing_page, converted]
Index: []
```

1.0.4 ToDo 1.4

Use **df2** in the cells below to answer the quiz questions related to **Quiz 4** in the classroom.
a. What is the probability of an individual converting regardless of the page they receive?

In [99]:
```
# Probability of converting regardless of the page recieved (old or new)
P_population = df2.converted.mean()
P_population
```

Out[99]: 0.11959708724499628

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

In [100]:
```
# Probability of a person in the control group converting
conv_c = df2.query("group == 'control' and converted == 1").shape[0] / df2.group[df2.g
conv_c
```

Out[100]: 0.1203863045004612

c. Given that anindividual was in the `treatment` group, what is the probability they converted?

In [101]:
```
# Probability of a person in the treatment group converting
conv_t = df2.query("group == 'treatment' and converted == 1").shape[0] / df2.group[df2
conv_t
```

Out[101]: 0.11880806551510564

In [102]:
```
# Calculate the actual difference (obs_diff) between the conversion rates for the two
df_c = df2.query("group == 'control'")
df_t = df2.query("group == 'treatment'")

obs_diff = df_t.converted.mean() - df_c.converted.mean()
obs_diff
```

```
Out[102]: -0.0015782389853555567
```

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

```
In [103]: # Probability of a person getting the new page
          df_t.shape[0] / df2.shape[0]
```

```
Out[103]: 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.

We can infer that the new page doesn't lead to more conversions as the evidence isn't sufficient to draw that conclusion.

However the test was certainly well designed, as 50% of the population got the old page and the other 50% received the new page (population size = 290584).

From the performed analysis, it is fairly clear there isn't any considerable difference in conversions, neither the old_page (12.04%) compared to the the new_page (11.88%).

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.

1.0.5 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.

Null Hypothesis (H0) = P(old) >= P(new),

Alternative Hypothesis (H1) = P(new) > P(old)

1.0.6 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.

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

```
In [104]: # null hypothesis states both new and old have the same conversion rates
          P_new = df2.converted.mean()
          P_new

Out[104]: 0.11959708724499628
```

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

```
In [105]: # null hypothesis states both new and old have the same conversion rates
          P_old = df2.converted.mean()
          P_old

Out[105]: 0.11959708724499628
```

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

```
In [106]: # Number of people in the treatment group
          n_new = df_t.shape[0]
          n_new

Out[106]: 145310
```

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

```
In [107]: # Number of people in the control group
          n_old = df_c.shape[0]
          n_old

Out[107]: 145274
```

e. Simulate Sample for the `treatment` Group Simulate n_{new} transactions with a conversion rate of p_{new} under the null hypothesis.

```
In [108]: # Simulated transactions with a conversion rate of 'P new' under H0
          new_page_converted= np.random.choice([1, 0], size = n_new, p = [P_new, (1-P_new)])
          new_page_converted.mean()

Out[108]: 0.11909710274585369
```

```
In [109]: # Simulate a Sample for the treatment Group
          sample_t = np.random.choice([1, 0], size = n_new, p = [P_new, (1-P_new)])
          sample_t.mean()

Out[109]: 0.11968894088500448
```

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 [110]: # Simulate a Sample for the control Group
          sample_c = np.random.choice([1, 0], size = n_old, p = [P_old, (1-P_old)])
          sample_c.mean()

Out[110]: 0.12001459311370238
```

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

```
In [111]: # Calculating the P_diff (difference in simulated conversion probability)
          P_diff = sample_t.mean() - sample_c.mean()
          P_diff

Out[111]: -0.00032565222869790356
```

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 [112]: # Sampling distribution
          p_diffs = []

          for _ in range(10000):
              sample_t = np.random.choice([1, 0], size = n_new, p = [P_new, (1-P_new)])
              mean_t = sample_t.mean()
              sample_c = np.random.choice([1, 0], size = n_old, p = [P_old, (1-P_old)])
              mean_c = sample_c.mean()
              p_diff = mean_t - mean_c
              p_diffs.append(p_diff)
```

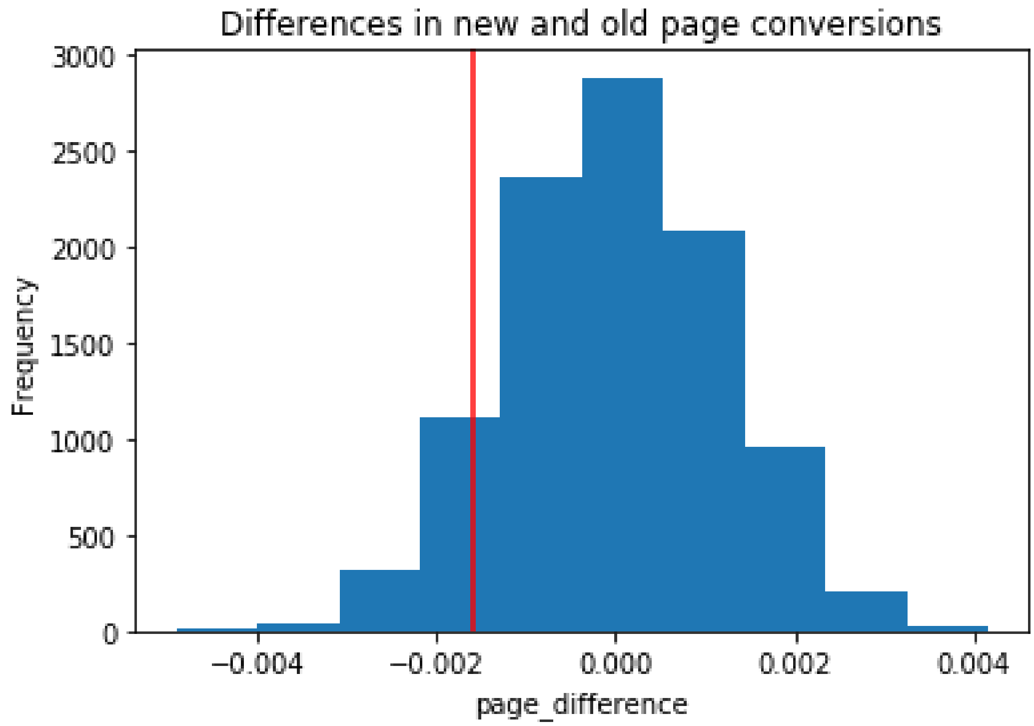
i. Histogram Plot a histogram of the **p_diffs**. Does this plot look like what you expected? Use the matching problem in the classroom to assure you fully understand what was computed here.

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

```
In [113]: p_diffs = np.array(p_diffs)

          plt.hist(p_diffs); # p_diffs histogram
```

```
plt.xlabel('page_difference') # x-label
plt.ylabel('Frequency') # y-label
plt.title('Differences in new and old page conversions') # Title of graph
plt.axvline(obs_diff, c='r');
```



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

```
In [114]: (p_diffs > obs_diff).mean()
```

```
Out[114]: 0.9052999999999999
```

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? *Hint*: Compare the value above with the "Type I error rate (0.05)".

This value in scientific studies is refered to as the p-value, which generally signifies that whether the comparison is statistically significant or not. From the results we are unable to reject the null hypothesis.

l. 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 walk-through 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 [115]: import statsmodels.api as sm

          # number of conversions with the old_page
          convert_old = df2.query("converted == 1 and landing_page == 'old_page'").shape[0]

          # number of conversions with the new_page
          convert_new = df2.query("converted == 1 and landing_page == 'new_page'").shape[0]

          # 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]

          # printed values for the above variables
          print(convert_old, convert_new, n_old, n_new)
```

```
17489 17264 145274 145310
```

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.

```
In [116]: # ToDo: Complete the sm.stats.proportions_ztest() method arguments
          z_score, p_value = sm.stats.proportions_ztest([convert_old,convert_new], [n_old, n_new
          print(z_score, p_value)
```

```
1.31092419842 0.094941687241
```

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.**?

The z-score coincides with the result from the p-value calculated earlier (z-score<95%).Therefore we are unable to reject the null hypothesis.

Part III - A regression approach

1.0.7 ToDo 3.1

In this final part, you will see that the result you achieved in the A/B test in Part II above can also be achieved by performing regression.

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 need to use logistice regression.

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 [117]: # intercept coloumn
          df2['intercept'] = 1

          # Creat dummy variables in a coloumn called 'ab_page'
          df2['ab_page'] = pd.get_dummies(df2['group'])['treatment']

          df2.head()
```

```
Out[117]:    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

             intercept  ab_page
          0          1        0
          1          1        0
          2          1        1
          3          1        1
          4          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 [118]: l_m = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])
          result = l_m.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 [119]: result.summary2()
```

```
Out[119]: <class 'statsmodels.iolib.summary2.Summary'>
          """
```

```
                        Results: Logit
=================================================================
Model:              Logit            No. Iterations:   6.0000
Dependent Variable: converted        Pseudo R-squared: 0.000
Date:               2022-09-22 02:15 AIC:              212780.3502
No. Observations:   290584           BIC:              212801.5095
Df Model:           1                Log-Likelihood:   -1.0639e+05
Df Residuals:       290582           LL-Null:          -1.0639e+05
Converged:          1.0000           Scale:            1.0000
------------------------------------------------------------------
              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**?

We can see that the p-value of the ab-page is 0.189 which is much higher than the type I error 0.05. So from this we can gather that we are unable to reject the null hypothesis, meaning that the new page isn't doing better than the old one in regards to conversion.

From part II the new page has a higher conversion rate than the old page meaning that it is one-sided.

Part III is two-sided as the new page can affect the conversion rate either positively or negatively. Results from part II & III are also unable to reject the null hypothesis.

Difference in p-values is due to 2 different hypothesis being tested in both of the tests.

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?

It will be good to add other variable to identify other potential factors affecting conversion rates.

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 [120]: # Read the countries.csv
          c_df = pd.read_csv('countries.csv')
          c_df.head()
```

Out[120]:
```
   user_id country
0   834778      UK
1   928468      US
2   822059      UK
3   711597      UK
4   710616      UK
```

In [121]:
```
# Join with the df2 dataframe
df_merged = c_df.set_index('user_id').join(df2.set_index('user_id'))
df_merged.head()
```

Out[121]:
```
        country                   timestamp      group landing_page  \
user_id
834778       UK  2017-01-14 23:08:43.304998    control     old_page
928468       US  2017-01-23 14:44:16.387854  treatment     new_page
822059       UK  2017-01-16 14:04:14.719771  treatment     new_page
711597       UK  2017-01-22 03:14:24.763511    control     old_page
710616       UK  2017-01-16 13:14:44.000513  treatment     new_page

         converted  intercept  ab_page
user_id
834778           0          1        0
928468           0          1        1
822059           1          1        1
711597           0          1        0
710616           0          1        1
```

In [122]:
```
# Create the necessary dummy variables
df_merged[['UK', 'US', 'CA']] = pd.get_dummies(df_merged['country'])
df_merged.head()
```

Out[122]:
```
        country                   timestamp      group landing_page  \
user_id
834778       UK  2017-01-14 23:08:43.304998    control     old_page
928468       US  2017-01-23 14:44:16.387854  treatment     new_page
822059       UK  2017-01-16 14:04:14.719771  treatment     new_page
711597       UK  2017-01-22 03:14:24.763511    control     old_page
710616       UK  2017-01-16 13:14:44.000513  treatment     new_page

         converted  intercept  ab_page  UK  US  CA
user_id
834778           0          1        0   0   1   0
928468           0          1        1   0   0   1
822059           1          1        1   0   1   0
711597           0          1        0   0   1   0
710616           0          1        1   0   1   0
```

In [123]:
```
l_m = sm.Logit(df_merged['converted'], df_merged[['intercept', 'US', 'CA', 'ab_page']]
results = l_m.fit()
results.summary2()
```

```
Optimization terminated successfully.
         Current function value: 0.366113
         Iterations 6
```

Out[123]:
```
<class 'statsmodels.iolib.summary2.Summary'>
"""
                          Results: Logit
==================================================================
Model:              Logit            No. Iterations:   6.0000
Dependent Variable: converted        Pseudo R-squared: 0.000
Date:               2022-09-22 02:15 AIC:              212781.1253
No. Observations:   290584           BIC:              212823.4439
Df Model:           3                Log-Likelihood:   -1.0639e+05
Df Residuals:       290580           LL-Null:          -1.0639e+05
Converged:          1.0000           Scale:            1.0000
-------------------------------------------------------------------
               Coef.  Std.Err.    z     P>|z|   [0.025   0.975]
-------------------------------------------------------------------
intercept     -2.0300   0.0266 -76.2488 0.0000 -2.0822 -1.9778
US             0.0506   0.0284   1.7835 0.0745 -0.0050  0.1063
CA             0.0408   0.0269   1.5161 0.1295 -0.0119  0.0934
ab_page       -0.0149   0.0114  -1.3069 0.1912 -0.0374  0.0075
==================================================================

"""
```

In [124]:
```python
# We need to make the coefficients more easily interpreted
np.exp(0.0506), np.exp(0.0408)
```

Out[124]:
```
(1.0519020483004984, 1.0416437559600236)
```

For people that live in the UK, they are 1.052 times more likely than ones living in the US, while holding all other variables constant.

For people that live in the UK, they are 1.042 times more likely than ones living in CA, while holding all other variables constant.

The p-value is above the error threshold (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 [125]:
```python
# Fit your model, and summarize the results
df_merged['US_ab_page'] = df_merged['US'] * df_merged['ab_page']
df_merged['CA_ab_page'] = df_merged['CA'] * df_merged['ab_page']

l_m = sm.Logit(df_merged['converted'], df_merged[['intercept', 'US', 'CA', 'ab_page',
```

```
results = l_m.fit()
results.summary2()
```

```
Optimization terminated successfully.
         Current function value: 0.366109
         Iterations 6
```

Out[125]:
```
<class 'statsmodels.iolib.summary2.Summary'>
"""
                          Results: Logit
==================================================================
Model:              Logit            No. Iterations:   6.0000
Dependent Variable: converted        Pseudo R-squared: 0.000
Date:               2022-09-22 02:15 AIC:              212782.6602
No. Observations:   290584           BIC:              212846.1381
Df Model:           5                Log-Likelihood:   -1.0639e+05
Df Residuals:       290578           LL-Null:          -1.0639e+05
Converged:          1.0000           Scale:            1.0000
-------------------------------------------------------------------
               Coef.   Std.Err.     z      P>|z|    [0.025   0.975]
-------------------------------------------------------------------
intercept     -2.0040    0.0364  -55.0077  0.0000  -2.0754  -1.9326
US             0.0118    0.0398    0.2957  0.7674  -0.0663   0.0899
CA             0.0175    0.0377    0.4652  0.6418  -0.0563   0.0914
ab_page       -0.0674    0.0520   -1.2967  0.1947  -0.1694   0.0345
US_ab_page     0.0783    0.0568    1.3783  0.1681  -0.0330   0.1896
CA_ab_page     0.0469    0.0538    0.8718  0.3833  -0.0585   0.1523
==================================================================

"""
```

In [126]:
```
np.exp(0.0783), np.exp(0.0469)
```

Out[126]:
```
(1.0814470441230692, 1.0480172021191829)
```

- Users in the UK treatment group are 1.082 times more likely to convert than users living in the US.
- Users in the UK treatment group are 1.048 times more likely to convert than users living in CA.
- Country of origin for the user does not seem to be significant in regards to conversion rate. Since none of the p-values are close to 0.05 (ie above error threshold). Therefore it is my opinion to not keep using the new page, as there does not seem to be any additional benefits.

Conclusion

- The results from the A/B testing and the regression model both show that the new page aren't able to get more than the old one. Therefore we are unable to reject the null hypothesis and my suggestion would be to stop using the new page and keep using the old one

Final Check!

Congratulations! You have reached the end of the A/B Test Results project! You should be very proud of all you have accomplished!

Submission You may either submit your notebook through the "SUBMIT PROJECT" button at the bottom of this workspace, or you may work from your local machine and submit on the last page of this project lesson.

1. Before you submit your project, you need to create a .html or .pdf version of this notebook in the workspace here. To do that, run the code cell below. If it worked correctly, you should get a return code of 0, and you should see the generated .html file in the workspace directory (click on the orange Jupyter icon in the upper left).

2. Alternatively, you can download this report as .html via the **File** > **Download as** submenu, and then manually upload it into the workspace directory by clicking on the orange Jupyter icon in the upper left, then using the Upload button.

3. Once you've done this, you can submit your project by clicking on the "Submit Project" button in the lower right here. This will create and submit a zip file with this .ipynb doc and the .html or .pdf version you created. Congratulations!

```
In [127]: from subprocess import call
          call(['python', '-m', 'nbconvert', 'Analyze_ab_test_results_notebook.ipynb'])
```

```
Out[127]: 0
```