Analyze_ab_test_results_notebook

October 26, 2021

0.1 Analyze A/B Test Results

You may either submit your notebook through the workspace here, or you may work from your local machine and submit through the next page. Either way assure that your code passes the project RUBRIC. **Please save regularly.**

This project will assure you have mastered the subjects covered in the statistics lessons. The hope is to have this project be as comprehensive of these topics as possible. Good luck!

0.2 Table of Contents

Introduction

A/B tests are very commonly performed by data analysts and data scientists. It is important that you get some practice working with the difficulties of these

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 page, keep the old page, or perhaps run the experiment longer to make their decision.

As you work through this notebook, follow along in the classroom and answer the corresponding quiz questions associated with each question. The labels for each classroom concept are provided for each question. This will assure you are on the right track as you work through the project, and you can feel more confident in your final submission meeting the criteria. As a final check, assure you meet all the criteria on the RUBRIC.

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)
```

1. Now, read in the `ab_data.csv` data. Store it in `df`. **Use your dataframe to answer the questions in Quiz 1 of the classroom.**

a. Read in the dataset 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.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 294478 entries, 0 to 294477
Data columns (total 5 columns):
user_id         294478 non-null int64
timestamp       294478 non-null object
group           294478 non-null object
landing_page    294478 non-null object
converted       294478 non-null int64
dtypes: int64(2), object(3)
memory usage: 11.2+ MB
```

c. The number of unique users in the dataset.

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

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

d. The proportion of users converted.

```
In [5]: df.query('converted == 1')['user_id'].nunique()/n
```

```
Out[5]: 0.12104245244060237
```

e. The number of times the `new_page` and `treatment` don't match.

```
In [6]: st1 = '(landing_page == "new_page" and group != "treatment")'
        st2 = '(landing_page != "new_page" and group == "treatment")'
        #print(st1+' or '+ st2)
        df.query(st1+' or '+ st2)['user_id'].nunique()
```

```
Out[6]: 3893
```

f. Do any of the rows have missing values?

```
In [7]: df.count()
```

```
Out[7]: user_id         294478
        timestamp       294478
        group           294478
        landing_page    294478
        converted       294478
        dtype: int64
```

No rows have missing values

2. For the rows where **treatment** does not match with **new_page** or **control** does not match with **old_page**, we cannot be sure if this row truly received the new or old page. Use **Quiz 2** in the classroom to figure out how we should handle these rows.

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 [8]: st1 = '(landing_page == "new_page" and group == "treatment")'
        st2 = '(landing_page != "new_page" and group != "treatment")'
        #print(st1+' or '+ st2)
        df2 = df.query(st1+' or '+ st2)
```

```
In [9]: # Double Check all of the correct rows were removed - this should be 0
        df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].sha
```

```
Out[9]: 0
```

3. Use **df2** and the cells below to answer questions for **Quiz3** in the classroom.

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

```
In [10]: df2['user_id'].nunique()
```

```
Out[10]: 290584
```

```
In [11]: df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 290585 entries, 0 to 294477
Data columns (total 5 columns):
user_id         290585 non-null int64
timestamp       290585 non-null object
group           290585 non-null object
landing_page    290585 non-null object
converted       290585 non-null int64
dtypes: int64(2), object(3)
memory usage: 13.3+ MB
```

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

```
In [12]: range_index = range(0,290585)
         df2['index'] = range_index
         df2.set_index('index',inplace=True)
         df2.head()
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
```

Out[12]:

index	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

```
In [13]: df2['user_id'].duplicated()[df2['user_id'].duplicated() == True]
```

```
Out[13]: index
         2862    True
         Name: user_id, dtype: bool
```

c. What is the row information for the repeat **user_id**?

```
In [14]: df2.iloc[2862]
```

```
Out[14]: user_id                         773192
         timestamp       2017-01-14 02:55:59.590927
         group                          treatment
         landing_page                    new_page
         converted                              0
         Name: 2862, dtype: object
```

d. Remove **one** of the rows with a duplicate **user_id**, but keep your dataframe as **df2**.

```
In [15]: df2 = df2.drop_duplicates(subset=['user_id'] , keep = 'first')
```

```
In [16]: df2[df2['user_id'] == 773192]
```

Out[16]:

index	user_id	timestamp	group	landing_page	converted
1876	773192	2017-01-09 05:37:58.781806	treatment	new_page	0

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 [17]: df2.head()
```

Out[17]:

index	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

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

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

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

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

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

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

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

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

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

```
In [21]: df2.query('landing_page =="new_page"').count()[0]/n
```

```
Out[21]: 0.50006194422266881
```

e. Consider your results from parts (a) through (d) above, and explain below whether you think there is sufficient evidence to conclude that the new treatment page leads to more conversions.

no, sufficient evidence since the probality of converting in the treatment group is the same as the probality as the probality of conversion in the control group

Part II - A/B Test

Notice that because of the time stamp associated with each event, you could technically run a hypothesis test continuously as each observation was observed.

However, then the hard question is 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. For now, consider you need to make the decision just based on all the data provided. 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 your null and alternative hypotheses be? You can state your hypothesis in terms of words or in terms of p_{old} and p_{new}, which are the converted rates for the old and new pages.

Put your answer here.

$H0 : p_{new} <= p_{old}$ $H1 : p_{new} > p_{old}$

2. Assume under the null hypothesis, p_{new} and p_{old} both have "true" success rates equal to the **converted** success rate regardless of page - that is p_{new} and p_{old} are equal. Furthermore, assume they are equal to the **converted** rate in **ab_data.csv** regardless of the page.

Use a sample size for each page equal to the ones in **ab_data.csv**.

Perform the sampling distribution for the difference in **converted** between the two pages over 10,000 iterations of calculating an estimate from the null.

Use the cells below to provide the necessary parts of this simulation. If this doesn't make complete sense right now, don't worry - you are going to work through the problems below to complete this problem. 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?

```
In [22]: n = df2.count()[0]
         p_old = df2.query('converted == 1').count()[0]/n
```

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

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

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

```
In [24]: n_new = df2.query('landing_page == "new_page"').count()[0]
```

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

```
In [25]: n_old = df2.query('landing_page == "old_page"').count()[0]
```

e. Simulate n_{new} transactions with a conversion rate of p_{new} under the null. Store these n_{new} 1's and 0's in **new_page_converted**.

```
In [26]: new_page_converted = np.random.binomial(n =1 , p = p_old ,size= n_new )
```

f. Simulate n_{old} transactions with a conversion rate of p_{old} under the null. Store these n_{old} 1's and 0's in **old_page_converted**.

```
In [27]: old_page_converted = np.random.binomial(n =1 , p = p_old ,size= n_old )
```

g. Find p_{new} - p_{old} for your simulated values from part (e) and (f).

```
In [28]: new_page_converted.mean()
```

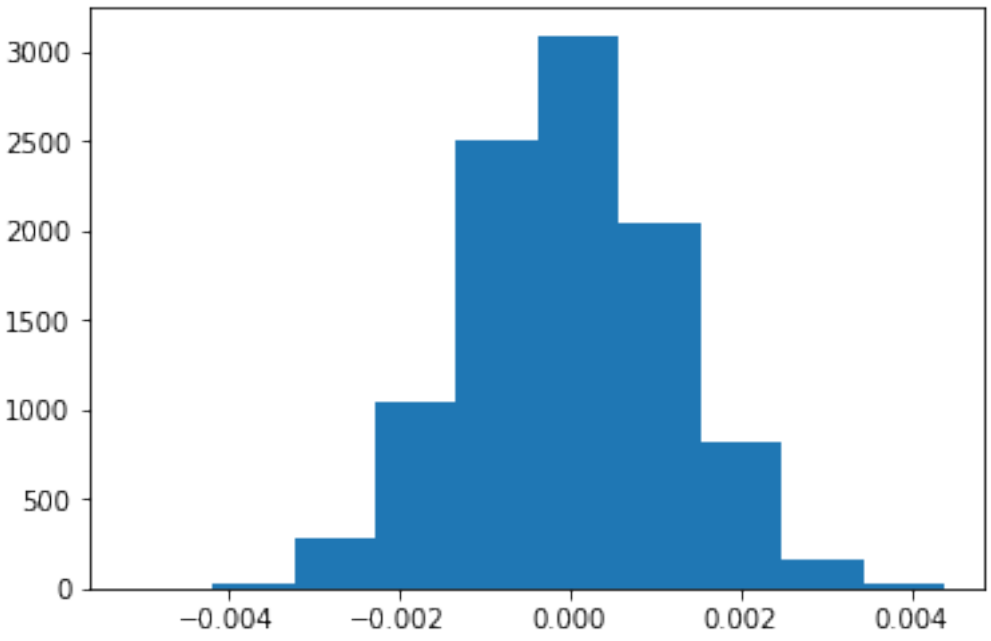
```
Out[28]: 0.11995733259927052

In [29]: pp = (new_page_converted.mean()) - (old_page_converted.mean())
         pp

Out[29]: 0.00047965593310865529
```

h. Create 10,000 p_{new} - p_{old} values using the same simulation process you used in parts (a) through (g) above. Store all 10,000 values in a NumPy array called **p_diffs**.

```
In [30]: p_diffs = []
         for _ in range (10000):
             new_page_converted = np.random.binomial(n =1 , p = p_old ,size= n_new )
             old_page_converted = np.random.binomial(n =1 , p = p_old ,size= n_old )
             pp = (new_page_converted.mean()) - (old_page_converted.mean())
             p_diffs.append(pp)
         p_diffs = np.array(p_diffs)
```

i. 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.

```
In [31]: plt.hist(p_diffs);
```

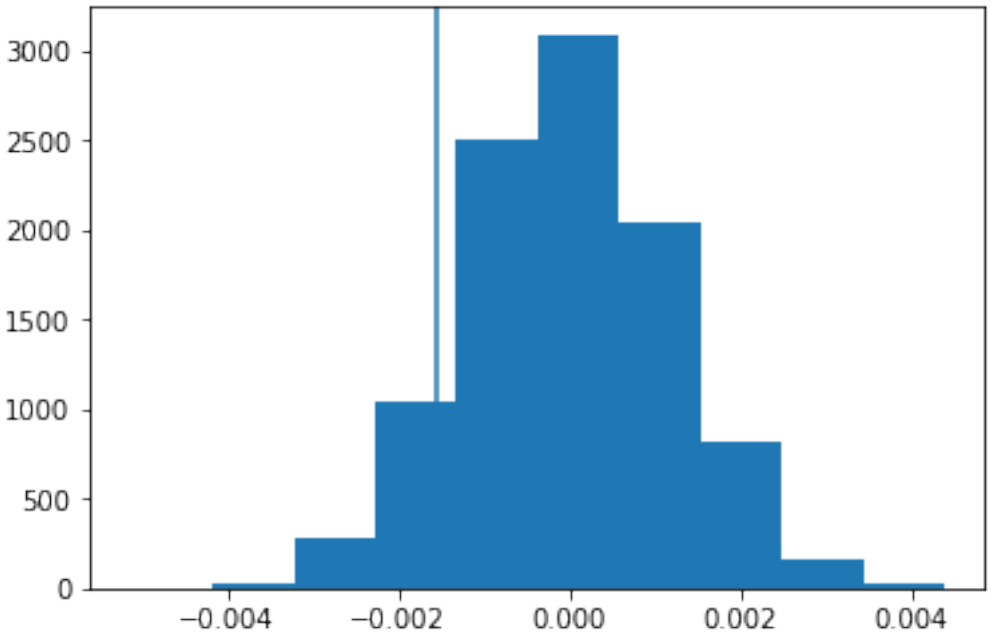



```
In [32]: p_diffs.mean()

Out[32]: -1.0217860152070251e-05
```

j. What proportion of the **p_diffs** are greater than the actual difference observed in **ab_data.csv**?

```
In [33]: ab_p_old = df2.query('landing_page =="old_page" and converted ==1').count()[0]/control_
         ab_p_new = df2.query('landing_page =="new_page" and converted ==1').count()[0]/control_
         difff = ab_p_new - ab_p_old
         plt.hist(p_diffs)
         plt.axvline(difff)
```

```
Out[33]: <matplotlib.lines.Line2D at 0x7fa5200657b8>
```




```
In [34]: p_value = (p_diffs > difff).mean()
         p_value , difff
```

```
Out[34]: (0.90159999999999996, -0.0015487974448284009)
```

k. Please explain using the vocabulary you've learned in this course what you just computed in part **j.** What is this value called in scientific studies? What does this value mean in terms of whether or not there is a difference between the new and old pages?

Put your answer here. Here we calculated the P-value in j which is equal to 0.8975 which as much greater than a significance level a(type I error) which is 0.05. Here we fail to reject the H_0, meaning there isn't a difference between new and old pages

l. 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 below to calculate the number of conversions for each page, as well as the number of individuals who received each page. Let `n_old` and `n_new` refer the the number of rows associated with the old page and new pages, respectively.

```
In [35]: import statsmodels.api as sm

         convert_old = df2.query('landing_page =="old_page" and converted ==1').count()[0]
         convert_new = df2.query('landing_page =="new_page" and converted ==1').count()[0]
         n_old = df2.query('landing_page =="old_page"').count()[0]
         n_new = df2.query('landing_page =="new_page"').count()[0]
         convert_old,convert_new,n_old,n_new
```

```
/opt/conda/lib/python3.6/site-packages/statsmodels/compat/pandas.py:56: FutureWarning: The panda
  from pandas.core import datetools
```

```
Out[35]: (17489, 17264, 145274, 145310)
```

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

```
In [36]: from statsmodels.stats.proportion import proportions_ztest
         N = np.array([n_new , n_old,])
         conversions = np.array([ convert_new ,convert_old ])
         stat, pval = proportions_ztest(count = conversions, nobs =N  , alternative ='larger')
         stat,pval
```

```
Out[36]: (-1.3109241984234394, 0.90505831275902449)
```

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.**?

Put your answer here. the P-value computed from the ztest is equal to the P-value we got from the sampling distribution therefore the agree with the findings

Part III - A regression approach

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 is either a conversion or no conversion, what type of regression should you be performing in this case?

Put your answer here. Logistic regression

b. The goal is to use **statsmodels** to fit the regression model you specified in part **a.** to see if there is a significant difference in conversion based on which page a customer receives. However, you first need to create in df2 a column for the intercept, and create a dummy variable column for which page each user received. Add an **intercept** column, as well as an **ab_page** column, which is 1 when an individual receives the **treatment** and 0 if **control**.

```
In [37]: df2.head()
```

Out[37]:

index	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

```
In [38]: df2['intercept'] =1

In [39]: df2 = df2.join(pd.get_dummies(df2['group']))#df['col_name'] = df['col_name'].map({5:1,

In [40]: df2['control'] = df2['control'].map({1:0,0:1})
         df2['treatment'] = df2['treatment'].map({1:0,0:1})

In [41]: df2.head()
```

Out[41]:

index	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	

index	intercept	control	treatment
0	1	0	1
1	1	0	1
2	1	1	0
3	1	1	0
4	1	0	1

```
In [42]: df2 = df2.rename(columns = {'control' : 'ab_page'})
         df2 = df2.drop(['treatment'] , axis = 1 )

In [43]: df2.head()
```

Out[43]:

index	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	

index	intercept	ab_page

```
index
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., then fit the model using the two columns you created in part **b.** to predict whether or not an individual converts.

```
In [44]: import statsmodels.api as sm;

In [45]:
         log = sm.Logit(df2['converted'] , df2[[  'intercept', 'ab_page' ]])
         results = log.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 [46]: results.summary2()

Out[46]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                 Results: Logit
         =================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2021-07-31 00:28 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**? **Hint**: What are the null and alternative hypotheses associated with your regression model, and how do they compare to the null and alternative hypotheses in **Part II**?

Put your answer here. the p-value associated with ab_page is 0.1899. The null associated with the regression model is $H_0 : p_{old} = p_{new}$, therefore the null and alternative hypothesis are not the same resulting in different p-values

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?

Put your answer here. considering other factors will lead to narroww down what variables that are have a significant result on the conversion rate however there is the problem of correlated errors which can make our model misleading

g. Now along with testing if the conversion rate changes for different pages, also add an effect based on which country a user lives in. You will need to read in the **countries.csv** dataset and merge together your datasets on the appropriate rows. Here are the docs for joining tables.

Does it appear that country had an impact on conversion? Don't forget to create dummy variables for these country columns - **Hint: You will need two columns for the three dummy variables.** Provide the statistical output as well as a written response to answer this question.

```
In [47]: countries = pd.read_csv('countries.csv')
         countries.head()
         joined_table = df2.merge(countries ,how = 'inner', on = 'user_id')
         joined_table.head()
```

```
Out[47]:    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 country
         0          1        0      US
         1          1        0      US
         2          1        1      US
         3          1        1      US
         4          1        0      US
```

```
In [48]: joined_table['country'].value_counts()
```

```
Out[48]: US    203619
         UK     72466
         CA     14499
         Name: country, dtype: int64
```

```
In [49]: joined_table[['CA', 'UK' , 'US']] = pd.get_dummies(joined_table['country'])
         joined_table.head()
```

Out[49]:

	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	country	CA	UK	US
0	1	0	US	0	0	1
1	1	0	US	0	0	1
2	1	1	US	0	0	1
3	1	1	US	0	0	1
4	1	0	US	0	0	1

In [50]:
```
log = sm.Logit(joined_table['converted'] , joined_table[[  'intercept', 'ab_page' , 'US
results = log.fit()
results.summary2()
```

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

Out[50]:
```
<class 'statsmodels.iolib.summary2.Summary'>
"""
                          Results: Logit
=================================================================
Model:              Logit            No. Iterations:   6.0000
Dependent Variable: converted        Pseudo R-squared: 0.000
Date:               2021-07-31 00:28 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
ab_page       -0.0149    0.0114   -1.3069  0.1912  -0.0374   0.0075
US             0.0408    0.0269    1.5161  0.1295  -0.0119   0.0934
UK             0.0506    0.0284    1.7835  0.0745  -0.0050   0.1063
=================================================================

"""
```

In [51]:
```
print(np.exp(0.0408) , np.exp(0.0506))
```

```
1.04164375596 1.0519020483
```

The coeffecients of the US and UK countries are 1.04 and 1.05 respectivaly compared to CA therefore US and UK conversions is 1.05 times the conversions of CA which is not statistically significant as well as having a P-value greater than 0.05 which make us fail to reject the null hypothesis, therefore countries **do not** have statistical significane on the conversion rate

h. 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 there significant effects on conversion. Create the necessary additional columns, and fit the new model.

Provide the summary results, and your conclusions based on the results.

```
In [52]: joined_table.head()
```

Out[52]:

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

```
In [53]: from sklearn.linear_model import LogisticRegression
         from patsy import dmatrices

In [54]: y, X = dmatrices('converted ~ ab_page + US + UK + ab_page:US + ab_page:UK', joined_tabl

In [55]: log = sm.Logit(y, X)
         results = log.fit()
         results.summary2()

Optimization terminated successfully.
         Current function value: 0.366109
         Iterations 6


Out[55]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                  Results: Logit
         =================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2021-07-31 00:28 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
ab_page       -0.0674    0.0520   -1.2967  0.1947  -0.1694   0.0345
US             0.0175    0.0377    0.4652  0.6418  -0.0563   0.0914
UK             0.0118    0.0398    0.2957  0.7674  -0.0663   0.0899
ab_page:US     0.0469    0.0538    0.8718  0.3833  -0.0585   0.1523
ab_page:UK     0.0783    0.0568    1.3783  0.1681  -0.0330   0.1896
==================================================================

"""
```

No statistcal significance although the P value decreased as well as the Coef. increased, the p value did not reach the 0.05 percent required to reject the null Hypothesis

1 Final Conclusion:

1.1 1)Probility:

the probility of conversion was basically the same for the new and old landing pages making it not statistically signifacant. ## 2)A/B test: The A/B test conducted produced a p value of 0.9 which is very much larger than our significance level (Type I error) making us fail to reject the null hypothesis ($H_0 : P_{new} <= P_{old}$)
3)Logistic Regression Model: the logistic regression model provided Coef values near 1 for categorical data and near 0 for quantitave data which does not reflect a relation between our dependant variable (**conversion**) and our independent variables which concluded (landing page, country, country x landing page).

2 Final Decision:

the new landing page **does NOT** have an impact on conversion rate.
Finishing Up

> Congratulations! You have reached the end of the A/B Test Results project! You should be very proud of all you have accomplished!
>
> **Tip**: Once you are satisfied with your work here, check over your report to make sure that it is satisfies all the areas of the rubric (found on the project submission page at the end of the lesson). You should also probably remove all of the "Tips" like this one so that the presentation is as polished as possible.

2.1 Directions to Submit

> 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).

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.

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 [56]:
```
from subprocess import call
call(['python', '-m', 'nbconvert', 'Analyze_ab_test_results_notebook.ipynb'])
```

Out[56]:
```
0
```