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!

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 [2]:
```
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 [3]:
```
df=pd.read_csv('ab_data.csv')
df.head(10)
```

Out[3]:

	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
5	936923	2017-01-10 15:20:49.083499	control	old_page	0
6	679687	2017-01-19 03:26:46.940749	treatment	new_page	1
7	719014	2017-01-17 01:48:29.539573	control	old_page	0
8	817355	2017-01-04 17:58:08.979471	treatment	new_page	1
9	839785	2017-01-15 18:11:06.610965	treatment	new_page	1

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

In [4]:
```
df.shape[0]
```

Out[4]: 294478

c. The number of unique users in the dataset.

In [5]:
```
df.user_id.nunique()
```

Out[5]: 290584

d. The proportion of users converted.

In [6]:
```
uniq_users=df.user_id.nunique()
df.query('converted==1').user_id.nunique()/uniq_users
```

Out[6]: 0.12104245244060237

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

In [7]:
```
len(df[(df['group']=='treatment') & (df['landing_page']!='new_page') | (df['group']!='treatm
ent') & (df['landing_page']=='new_page')])
```

Out[7]: 3893

f. Do any of the rows have missing values?

In [8]:
```
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
```

`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 [9]:
```
df_remove=df[(df['group']=='treatment') & (df['landing_page']!='new_page') | (df['group']!=
'treatment') & (df['landing_page']=='new_page')]
```

In [10]:
```
df2=df.drop((df_remove).index,inplace=False)
df2.shape[0]
```

Out[10]: 290585

In [11]:
```
# Double Check all of the correct rows were removed - this should be 0
df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].shape[0
]
```

Out[11]: 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 [12]:
```
df2.shape[0]
df2.user_id.nunique()
```

Out[12]: 290584

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

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

Out[13]:
```
2893    773192
Name: user_id, dtype: int64
```

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

In [14]:
```
duplicate_row=df2[df2['user_id']==773192]
duplicate_row
```

Out[14]:

	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**, but keep your dataframe as **df2**.

In [15]:
```
# Since the timestamp for both the entries are different, we need to filter
# the row using user_id column

df2.drop_duplicates(subset='user_id',inplace=True)
```

In [16]:
```
df2.drop_duplicates(subset='user_id',inplace=True)
```

In [17]:
```
df2[df2['user_id']==773192]
```

Out[17]:

	user_id	timestamp	group	landing_page	converted
1899	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 [20]:
```
# This also works fine but better use the minimized ones : len(df2[df2['converted']==1])/len
(df2['landing_page'])
p_converted = df2.converted.mean()
p_converted
```

Out[20]: 0.11959708724499628

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

In [21]:
```
len(df2[(df2['group']=='control') & (df2['converted']==1)])/len(df2[(df2['group']=='control'
)])
```

Out[21]: 0.1203863045004612

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

In [22]:
```
len(df2[(df2['group']=='treatment') & (df2['converted']==1)])/len(df2[(df2['group']=='treatm
ent')])
```

Out[22]: 0.11880806551510564

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

In [23]:
```
len(df2[df2['landing_page']=='new_page'])/len(df2['landing_page'])
```

Out[23]: 0.5000619442226688

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.

Your answer goes here. ***Ofcourse not! The results generated above rather suggest us that the control page conversion is higher than the treatment page. However, the results generated upon the above calculations are not enough to support that the control page conversions are better because, often these predictions are delusional.***

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_{old} = p_{new}

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 [21]:
```
df2.head(5)
```

Out[21]:

	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 [24]:
```
# Assume under the null hypothesis,  pnew  and  pold  both have "true" success
# rates equal to the converted success rate regardless of page - that is  pnew
# and  pold  are equal.
p_new=df2['converted'].mean()
p_new
```

Out[24]: 0.11959708724499628

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

In [25]:
```
p_old=df2['converted'].mean()
p_old
```

Out[25]: 0.11959708724499628

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

In [26]:
```
n_new=df2[df2['group']=='treatment'].count()
n_new
```

Out[26]:
```
user_id         145310
timestamp       145310
group           145310
landing_page    145310
converted       145310
dtype: int64
```

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

In [27]:
```
df2[df2['group']=='control'].count()
```

Out[27]:
```
user_id         145274
timestamp       145274
group           145274
landing_page    145274
converted       145274
dtype: int64
```

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 [28]:
```
new_page_converted=np.random.choice([0,1],size=145310,p=[1-p_new,p_new])
p_new_converted=new_page_converted.mean()
p_new_converted
```

Out[28]: 0.11864978322207693

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 [29]:
```
old_page_converted=np.random.choice([0,1],size=145274,p=[1-p_old,p_old])
p_old_converted=old_page_converted.mean()
p_old_converted
```

Out[29]: 0.12012472982089017

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

In [30]:
```
p_new_converted-p_old_converted
```

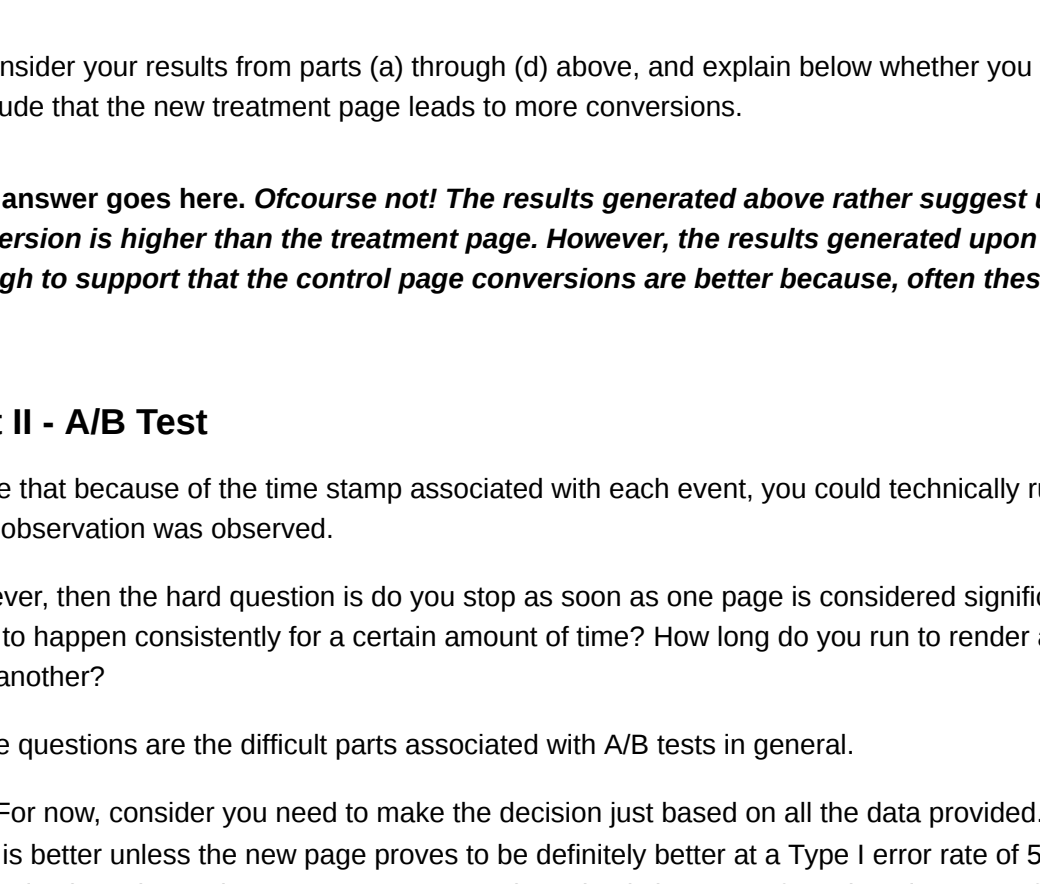
Out[30]: -0.0014749465988132399

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 [35]:
```
p_diffs=[]
for i in range(10000):

    new_page_converted=np.random.choice([0,1],size=145310,p=[1-p_new,p_new])
    p_new_converted=new_page_converted.mean()

    old_page_converted=np.random.choice([0,1],size=145274,p=[1-p_old,p_old])
    p_old_converted=old_page_converted.mean()

    p_diffs.append(p_new_converted-p_old_converted)
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 [36]:
```
plt.hist(p_diffs);
```



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

In [37]:
```
obs_diff=df2.query('group=="treatment"')['converted'].mean()-df2.query('group=="control"')[
'converted'].mean()
obs_diff
```

Out[37]: -0.0015782389853555567

In [38]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 290584 entries, 0 to 294477
Data columns (total 5 columns):
user_id         290584 non-null int64
timestamp       290584 non-null object
group           290584 non-null object
landing_page    290584 non-null object
converted       290584 non-null int64
dtypes: int64(2), object(3)
memory usage: 13.3+ MB
```

In [39]:
```
(p_diffs>obs_diff).mean()
```

Out[39]: 0.90229999999999999

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. The value in part 'j' is the 'p-value' that we obtain after comparing the mean of observed diference mean and the mean obtained by simulated difference between new and old converted pages(p_diffs).

Observed difference mean(obs_diff) is the what we get by observing the actual values from the dataset whereas p_diffs we get by simulating the transactions for 10000 times and then calculating the difference in means of new_page_converted and old_page_converted.

Now, By obs_diff being negative, it suggests us that the null hypothesis can't be rejected implying that old page is better to keep than implementing a new ones.

Secondly, the p-value being much much greater than 0.05 suggests the same that we should stick to Null hypothesis which states that the older page is better than the new ones.

Moreover, we did not build another histogram here to simulate distribution under null hypothesis unlike before due to the fact that, p_new and p_old are calculated assuming to be under null hypothesis.

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 [40]:
```
import statsmodels.api as sm

convert_old = len(df2[(df2['landing_page']=='old_page') & (df2['converted']==1)])
convert_new = len(df2[(df2['landing_page']=='new_page') & (df2['converted']==1)])
n_old = len(df2.query('landing_page=="old_page"'))
n_new = len(df2.query('landing_page=="new_page"'))
```

```
/opt/conda/lib/python3.6/site-packages/statsmodels/compat/pandas.py:56: FutureWarning: The pa
ndas.core.datetools module is deprecated and will be removed in a future version. Please use
the pandas.tseries module instead.
  from pandas.core import datetools
```

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 [41]:
```
from scipy.stats import norm
z_score, p_value = sm.stats.proportions_ztest([convert_old, convert_new], [n_old, n_new], al
ternative="smaller")
print(z_score)
print(p_value)
```

```
1.31092419842
0.905058312759
```

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.

Yes, they agree to the findings in j and k We know that when a p-value is greater than the value of alpha that is 0.05, we fail to reject the Null hypothesis.

Also, considering the z-score, if the z-score lies between -1.96 and +1.96 (1.31 here) then, the p-value is likely to be larger than 0.05(0.905 here) implies that you can't reject the null hypothesis.

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.

Undoubtedly, 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 [42]:
```
df2.head(4)
```

Out[42]:

	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

In [43]:
```
df2.head(5)
```

Out[43]:

	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 [44]:
```
df2['intercept'] = 1
df2['ab_page'] = pd.get_dummies(df2['group'])['treatment']
```

In [47]:
```
df2.head(5)
```

Out[47]:

	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

In [48]:
```
df2.head(6)
```

Out[48]:

	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
5	936923	2017-01-10 15:20:49.083499	control	old_page	0	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 [49]:
```
import statsmodels.api as sm
df2['intercept']=1
log_mode=sm.Logit(df2['converted'],df2[['intercept','ab_page']])
```

In [50]:
```
results=log_mode.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 [56]:
```
from scipy import stats
stats.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df)
results.summary()
```

Out[56]:

Logit Regression Results

Dep. Variable:	converted	No. Observations:	290584
Model:	Logit	Df Residuals:	290581
Method:	MLE	Df Model:	2
Date:	Sun, 03 May 2020	Pseudo R-squ.:	1.521e-05
Time:	05:04:56	Log-Likelihood:	-1.0639e+05
converged:	True	LL-Null:	-1.0639e+05
		LLR p-value:	0.1984

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	-2.0375	0.026	-78.364	0.000	-2.088	-1.987
UK	0.0507	0.028	1.786	0.074	-0.005	0.106
US	0.0408	0.027	1.518	0.129	-0.012	0.093

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.

Both the p_values obtained differ from one another. The pvalue obtained in part 2 was 0.905 whereas, the p-value obtained in part-3 is 0.19. The difference in the p-values is due to using two different testing approaches.

In part-2 we used, one-tailed test where it was concerned only about guessing whether the new page was better.

On the other hand, in part-3, two-tailed test was performed which focuses on whether there is a significant difference in conversion between control and treatment.

H1 : p_{new} != p_{old}

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.

The age factor could help us give an insight about the pattern of ages that are involved in the conversion. Also, the gender factor would serve as a helping hand in carrying out the analysis to know whether the rate of males are involving more for the conversion or women.

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 [51]:
```
df_countries=pd.read_csv('countries.csv')
df_countries.head()
```

Out[51]:

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

In [52]:
```
df_merge=pd.merge(df2,df_countries,on='user_id')
df_merge['country'].value_counts()
```

Out[52]:
```
US    203619
UK     72466
CA     14499
Name: country, dtype: int64
```

In [53]:
```
df_merge.tail(10)
```

Out[53]:

	user_id	timestamp	group	landing_page	converted	intercept	ab_page	country
290574	643562	2017-01-02 19:20:05.460595	treatment	new_page	0	1	1	CA
290575	755438	2017-01-18 17:35:06.149568	control	old_page	0	1	0	US
290576	908354	2017-01-11 02:42:21.195145	control	old_page	0	1	0	US
290577	718310	2017-01-21 22:44:20.378320	control	old_page	0	1	0	US
290578	822004	2017-01-04 03:36:46.071379	treatment	new_page	0	1	1	CA
290579	751197	2017-01-03 22:28:38.630509	control	old_page	0	1	0	US
290580	945152	2017-01-12 00:51:57.078372	control	old_page	0	1	0	US
290581	734608	2017-01-22 11:45:03.439544	control	old_page	0	1	0	US
290582	697314	2017-01-15 01:20:28.957438	control	old_page	0	1	0	US
290583	715931	2017-01-16 12:40:24.467417	treatment	new_page	0	1	1	UK

In [54]:
```
df_countries[['UK', 'US']] = pd.get_dummies(df_countries['country'])[['UK','US']]
df_countries.head()
df_merge=pd.merge(df2,df_countries,on='user_id')
df_merge.head()
```

Out[54]:

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

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 [57]:
```
model=sm.Logit(df_merge['converted'],df_merge[['intercept','UK','US']])
results=model.fit()
results.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.366116
         Iterations 6
```

Out[57]:

Logit Regression Results

Dep. Variable:	converted	No. Observations:	290584
Model:	Logit	Df Residuals:	290581
Method:	MLE	Df Model:	2
Date:	Sun, 03 May 2020	Pseudo R-squ.:	1.521e-05
Time:	05:05:06	Log-Likelihood:	-1.0639e+05
converged:	True	LL-Null:	-1.0639e+05
		LLR p-value:	0.1984

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	-2.0375	0.026	-78.364	0.000	-2.088	-1.987
UK	0.0507	0.028	1.786	0.074	-0.005	0.106
US	0.0408	0.027	1.518	0.129	-0.012	0.093

From the summary statistics , we can clearly see that the p-values are greater than 0.05 which suggests that the Null hypothesis can't be rejected. But, we already have had enough results which suggested us the same. So, basically adding the country column has not provided us with any significant information and henceforth, there is no significant impact on adding the countries column

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 [58]:
```
df_merge[['US_ab_page','UK_ab_page']]=pd.get_dummies(df_merge['ab_page'])
```

In [59]:
```
df_merge.head(4)
```

Out[59]:

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

In [72]:
```
log_mode=sm.Logit(df_merge['converted'],df_merge[['intercept','US','UK','US_ab_page','UK_ab_
page']])
results=log_mode.fit()
results.summary()
```

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.366113
         Iterations: 35
```

```
/opt/conda/lib/python3.6/site-packages/statsmodels/base/model.py:496: ConvergenceWarning: Max
imum Likelihood optimization failed to converge. Check mle_retvals
  "Check mle_retvals", ConvergenceWarning)
```

Out[72]:

Logit Regression Results

Dep. Variable:	converted	No. Observations:	290584
Model:	Logit	Df Residuals:	290580
Method:	MLE	Df Model:	3
Date:	Sun, 03 May 2020	Pseudo R-squ.:	2.323e-05
Time:	02:03:11	Log-Likelihood:	-1.0639e+05
converged:	False	LL-Null:	-1.0639e+05
		LLR p-value:	0.1760

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	-1.3583	7.04e+05	-1.93e-06	1.000	-1.38e+06	1.38e+06
US	0.0408	0.027	1.516	0.130	-0.012	0.093
UK	0.0506	0.028	1.784	0.074	-0.005	0.106