Analyze A/B Test Results

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 [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')
```

In [3]:
```
df.head()
```

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

b. Use the below cell 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]:
```
df['converted'].mean() * 100
```

Out[6]:
```
11.96591935560551
```

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

In [7]:
```
df.groupby(["group", "landing_page"]).size()
```

Out[7]:
```
group      landing_page
control    new_page          1928
           old_page        145274
treatment  new_page        145311
           old_page          1965
dtype: int64
```

In [8]:
```
df.query("group == 'treatment' and landing_page == 'old_page'").shape[0] + df.query("group == 'control'
 and landing_page == 'new_page'").shape[0]
```

Out[8]:
```
3893
```

f. Do any of the rows have missing values?

In [9]:
```
df.info()
```

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

`2.` For the rows where **treatment** is not aligned with **new_page** or **control** is not aligned with **old_page**, we cannot be sure if this row truly received the new or old page. Use **Quiz 2** in the classroom to provide 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 [10]:
```
df2 = df.query("group == 'control' and landing_page == 'old_page'")
df2 = df2.append(df.query("group == 'treatment' and landing_page == 'new_page'"))
df2.head()
```

Out[10]:

	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
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
7	719014	2017-01-17 01:48:29.539573	control	old_page	0

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['user_id'].nunique()
```

Out[12]:
```
290584
```

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

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

Out[13]:
```
1
```

In [14]:
```
df2[df2['user_id'].duplicated()]
```

Out[14]:

	user_id	timestamp	group	landing_page	converted
2893	773192	2017-01-14 02:55:59.590927	treatment	new_page	0

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

In [15]:
```
df2[df2['user_id'] == 773192]
```

Out[15]:

	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 [16]:
```
df2 = df2.drop_duplicates('user_id')
```

In [17]:
```
df2['user_id'].duplicated().sum()
```

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

`4.` Use **df2** in the below cells 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 [18]:
```
df2.converted.mean()
```

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

P(converted) = 0.1196

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

In [19]:
```
df2.query("group == 'control'")['converted'].mean()
```

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

P(converted | control) = 0.1204

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

In [20]:
```
df2.query("group == 'treatment'")['converted'].mean()
```

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

P(converted | treatment) = 0.1188

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

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

Out[21]:
```
0.5000619442226688
```

P(new page) = 0.5001

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

Not sufficient.

The new treatment page `P(converted | treatment) = 0.1188` leads to lower conversions rate than the old control page `P(converted | control) = 0.1204`. but the difference appears to be negligible and not sufficient evidence.

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.

H_0 : **p_{new} $<=$ p_{old}**

H_1 : **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 **convert rate** for p_{new} under the null?

```
In [22]: p_new = df2['converted'].mean()
         p_new
```

```
Out[22]: 0.11959708724499628
```

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

```
In [23]: p_old = df2['converted'].mean()
         p_old
```

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

c. What is n_{new}?

```
In [24]: n_new = df2.query("group == 'treatment'").shape[0]
         n_new
```

```
Out[24]: 145310
```

d. What is n_{old}?

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

```
Out[25]: 145274
```

e. Simulate n_{new} transactions with a convert 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_new, p_new)
         new_page_converted
```

```
Out[26]: 17374
```

f. Simulate n_{old} transactions with a convert 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_old, p_old)
         old_page_converted
```

```
Out[27]: 17319
```

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

```
In [28]: (new_page_converted / n_new) - (old_page_converted / n_old)
```

```
Out[28]: 0.00034896579952159446
```

h. Simulate 10,000 p_{new} - p_{old} values using this same process similarly to the one you calculated in parts **a. through g.** above. Store all 10,000 values in a numpy array called **p_diffs**.

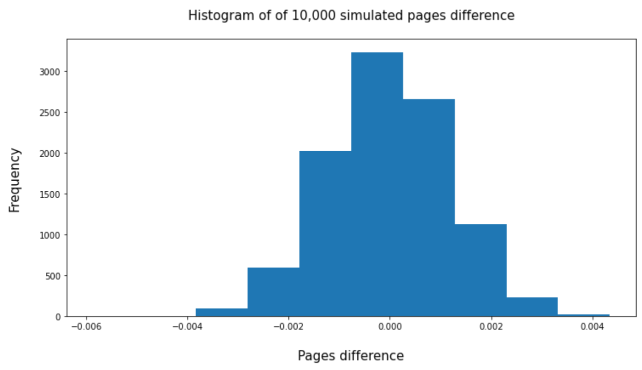
```
In [29]: p_diffs = (np.random.binomial(n_new, p_new, 10000) / n_new) - (np.random.binomial(n_old, p_old, 10000)
         / n_old)
         p_diffs = np.array(p_diffs)
         p_diffs
```

```
Out[29]: array([-1.08968471e-03,  1.10616589e-03, -1.48880237e-03, ...,
                 2.72999534e-04, -1.87827861e-04,  1.84192482e-05])
```

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 [30]: plt.figure(figsize=(12,6))
         plt.hist(p_diffs);
         plt.title('\nHistogram of of 10,000 simulated pages difference\n',fontsize=15)
         plt.xlabel('\nPages difference', fontsize=15)
         plt.ylabel('Frequency\n', fontsize=15)
```

```
Out[30]: Text(0, 0.5, 'Frequency\n')
```



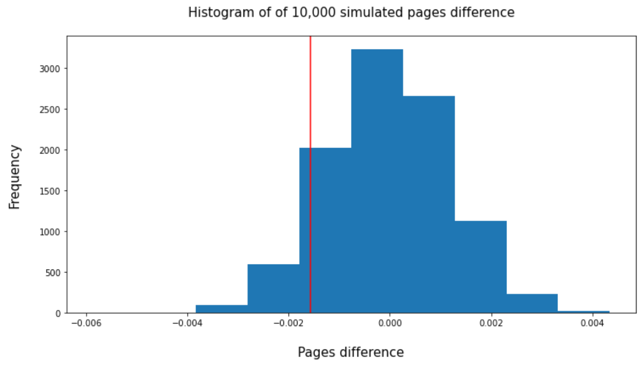
j. What proportion of the **p_diffs** are greater than the actual difference observed in **ab_data.csv**?

```
In [31]: obs_diff = df2.query("group == 'treatment'")['converted'].mean() - df2.query("group == 'control'")['con
         verted'].mean()
         (p_diffs > obs_diff).mean()
```

```
Out[31]: 0.9029
```

```
In [32]: plt.figure(figsize=(12,6))
         plt.hist(p_diffs);
         plt.title('\nHistogram of of 10,000 simulated pages difference\n',fontsize=15)
         plt.xlabel('\nPages difference', fontsize=15)
         plt.ylabel('Frequency\n', fontsize=15)
         plt.axvline(x = obs_diff, color = 'red')
```

```
Out[32]: <matplotlib.lines.Line2D at 0x286142bfb38>
```



k. In words, explain 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?

- What is this value called in scientific studies? **The p-value is the probability of obtaining a statistic as extreme or more extreme than the one observed in the experiment.**
- What does this value mean in terms of whether or not there is a difference between the new and old pages? **A large p-value in this case indicates that there is a slightly larger conversion rate for the new treatment page than the old control page.**

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 [33]: import statsmodels.api as sm

         convert_old = df2.query("group == 'control'")['converted'].sum()
         convert_new = df2.query("group == 'treatment'")['converted'].sum()
         n_old = df2.query("group == 'control'").shape[0]
         n_new = df2.query("group == 'treatment'").shape[0]
```

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 [34]:

```
z_score, p_value = sm.stats.proportions_ztest([convert_old, convert_new], [n_old, n_new], alternative=
'smaller')
(z_score, p_value)
```

Out[34]:

```
(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.**?

In [35]:

```
from scipy.stats import norm
norm.cdf(z_score)
```

Out[35]:

```
0.9050583127590245
```

In [36]:

```
norm.ppf(1-(0.05))
```

Out[36]:

```
1.6448536269514722
```

- 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?
  The `z-score = 1.31` is inside our `critical value = 1.64` and the `p-value = 0.905` is still large.
  So the null hypothesis H_0 that the conversion rate of the old_pages is higher than the conversion rate of the new_pages is not rejected.
- Do they agree with the findings in parts **j.** and **k.**? **These values agree with the findings in parts j. and k.**

Part III - A regression approach

`1.` In this final part, you will see that the result you acheived in the previous A/B test can also be acheived 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?

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 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]:

```
df3 = df.copy()
df3.head()
```

Out[37]:

	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]:

```
df3['intercept'] = 1
df3['ab_page'] = pd.get_dummies(df['group'])['treatment']
df3
```

Out[38]:

	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
...	...	...	...	...	...	...	...
294473	751197	2017-01-03 22:28:38.630509	control	old_page	0	1	0
294474	945152	2017-01-12 00:51:57.078372	control	old_page	0	1	0
294475	734608	2017-01-22 11:45:03.439544	control	old_page	0	1	0
294476	697314	2017-01-15 01:20:28.957438	control	old_page	0	1	0
294477	715931	2017-01-16 12:40:24.467417	treatment	new_page	0	1	1

294478 rows × 7 columns

c. Use **statsmodels** to import your regression model. Instantiate the model, and fit the model using the two columns you created in part **b.** to predict whether or not an individual converts.

In [39]:

```
import statsmodels.api as sm
logit = sm.Logit(df3['converted'],df3[['intercept','ab_page']])
results = logit.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.366243
         Iterations 6
```

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

In [40]:

```
results.summary()
```

Out[40]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	294478
Model:	Logit	**Df Residuals:**	294476
Method:	MLE	**Df Model:**	1
Date:	Sun, 11 Jul 2021	**Pseudo R-squ.:**	7.093e-06
Time:	19:08:15	**Log-Likelihood:**	-1.0785e+05
converged:	True	**LL-Null:**	-1.0785e+05
Covariance Type:	nonrobust	**LLR p-value:**	0.2161

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	-1.9887	0.008	-248.297	0.000	-2.004	-1.973
ab_page	-0.0140	0.011	-1.237	0.216	-0.036	0.008

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 the **Part II**?

- What is the p-value associated with **ab_page**? **0.216**
- Why does it differ from the value you found in **Part II**?
  The null and alternative hypotheses for the regression model are:
  H_0 : **p_{new} $ = $ p_{old}**
  H_1 : **p_{new} $!= $ p_{old}**
  This only predicts the difference in the two values and while Part II predicts which page gets more conversions rate.

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?

- **Other factors will help the regression model to get more accurate predictions of whether or not an individual converts.**
- **A disadvantage to adding additional terms into your regression model is that the model gets more complex and can cause bias in testing and interpreting.**

g. Now along with testing if the conversion rate changes for different pages, also add an effect based on which country a user lives. You will need to read in the **countries.csv** dataset and merge together your datasets on the approporiate 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 [41]:

```
countries_df = pd.read_csv('./countries.csv')
countries_df.head()
```

Out[41]:

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

In [42]:

```
df_new = countries_df.set_index('user_id').join(df2.set_index('user_id'), how='inner')
df_new.head()
```

Out[42]:

	country	timestamp	group	landing_page	converted
user_id					
834778	UK	2017-01-14 23:08:43.304998	control	old_page	0
928468	US	2017-01-23 14:44:16.387854	treatment	new_page	0
822059	UK	2017-01-16 14:04:14.719771	treatment	new_page	1
711597	UK	2017-01-22 03:14:24.763511	control	old_page	0
710616	UK	2017-01-16 13:14:44.000513	treatment	new_page	0

In [43]:

```
df_new['country'].value_counts()
```

Out[43]:

```
US    203619
UK     72466
CA     14499
Name: country, dtype: int64
```

In [44]:

```
### Create the necessary dummy variables
df_new['intercept'] = 1
df_new[['US', 'UK', 'CA']] = pd.get_dummies(df_new['country'])
df_new
```

Out[44]:

	country	timestamp	group	landing_page	converted	intercept	US	UK	CA
user_id									
834778	UK	2017-01-14 23:08:43.304998	control	old_page	0	1	0	1	0
928468	US	2017-01-23 14:44:16.387854	treatment	new_page	0	1	0	0	1
822059	UK	2017-01-16 14:04:14.719771	treatment	new_page	1	1	0	1	0
711597	UK	2017-01-22 03:14:24.763511	control	old_page	0	1	0	1	0
710616	UK	2017-01-16 13:14:44.000513	treatment	new_page	0	1	0	1	0
...	...	...	...	...	...	...	...	...	...
653118	US	2017-01-09 03:12:31.034796	control	old_page	0	1	0	0	1
878226	UK	2017-01-05 15:02:50.334962	control	old_page	0	1	0	1	0
799368	UK	2017-01-09 18:07:34.253935	control	old_page	0	1	0	1	0
655535	CA	2017-01-09 13:30:47.524512	treatment	new_page	0	1	1	0	0
934996	UK	2017-01-09 00:30:08.377677	control	old_page	0	1	0	1	0

290584 rows × 9 columns

In [45]:

```
### Fit Your Linear Model And Obtain the Results
logit = sm.Logit(df_new['converted'],df_new[['intercept','US', 'UK']])
results = logit.fit()
results.summary()
```

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

Out[45]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290584

Model:	Logit	**Df Residuals:**	290581
Method:	MLE	**Df Model:**	2
Date:	Sun, 11 Jul 2021	**Pseudo R-squ.:**	1.521e-05
Time:	19:08:16	**Log-Likelihood:**	-1.0639e+05
converged:	True	**LL-Null:**	-1.0639e+05
Covariance Type:	nonrobust	**LLR p-value:**	0.1984

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	-1.9967	0.007	-292.314	0.000	-2.010	-1.983
US	-0.0408	0.027	-1.518	0.129	-0.093	0.012
UK	0.0099	0.013	0.746	0.456	-0.016	0.036

It doesn't appear that country a user life affects the conversion rate because all the p_values are still larger than the Type I Error 0.05.

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 [46]:

```
df_new['ab_page'] = pd.get_dummies(df_new['group'])['treatment']
df_new
```

Out[46]:

	country	timestamp	group	landing_page	converted	intercept	US	UK	CA	ab_page
user_id										
834778	UK	2017-01-14 23:08:43.304998	control	old_page	0	1	0	1	0	0
928468	US	2017-01-23 14:44:16.387854	treatment	new_page	0	1	0	0	1	1
822059	UK	2017-01-16 14:04:14.719771	treatment	new_page	1	1	0	1	0	1
711597	UK	2017-01-22 03:14:24.763511	control	old_page	0	1	0	1	0	0
710616	UK	2017-01-16 13:14:44.000513	treatment	new_page	0	1	0	1	0	1
...	...	...	...	...	...	...	...	...	...	...
653118	US	2017-01-09 03:12:31.034796	control	old_page	0	1	0	0	1	0
878226	UK	2017-01-05 15:02:50.334962	control	old_page	0	1	0	1	0	0
799368	UK	2017-01-09 18:07:34.253935	control	old_page	0	1	0	1	0	0
655535	CA	2017-01-09 13:30:47.524512	treatment	new_page	0	1	1	0	0	1
934996	UK	2017-01-09 00:30:08.377677	control	old_page	0	1	0	1	0	0

290584 rows × 10 columns

In [47]:

```
### Fit Your Linear Model And Obtain the Results
logit = sm.Logit(df_new['converted'],df_new[['intercept','ab_page', 'US', 'UK']])
results = logit.fit()
results.summary()
```

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

Out[47]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290584
Model:	Logit	**Df Residuals:**	290580
Method:	MLE	**Df Model:**	3
Date:	Sun, 11 Jul 2021	**Pseudo R-squ.:**	2.323e-05
Time:	19:08:18	**Log-Likelihood:**	-1.0639e+05
converged:	True	**LL-Null:**	-1.0639e+05
Covariance Type:	nonrobust	**LLR p-value:**	0.1760

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	-1.9893	0.009	-223.763	0.000	-2.007	-1.972
ab_page	-0.0149	0.011	-1.307	0.191	-0.037	0.007
US	-0.0408	0.027	-1.516	0.130	-0.093	0.012
UK	0.0099	0.013	0.743	0.457	-0.016	0.036

It doesn't appear that interaction between the page and country affects the conversion rate because all the p_values are still larger than the Type I Error 0.05.