# 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 [55]:
```
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 below cell to find the number of rows in the dataset.

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

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

c. The number of unique users in the dataset.

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

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

d. The proportion of users converted.

In [5]:
```
df.converted.mean()*100
```

Out[5]:
```
11.96591935560551
```

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

In [6]:
```
df_q1 = df.query("group == 'treatment' and landing_page == 'old_page'").shape[0]
df_q2 = df.query("group == 'control' and landing_page == 'new_page'").shape[0]
df_q = df_q1 + df_q2
df_q
```

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

f. Do any of the rows have missing values?

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
user_id         0
timestamp       0
group           0
landing_page    0
converted       0
dtype: int64
```

`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 [8]:
```
df.drop(df.query("group == 'treatment' and landing_page == 'old_page'").index, inplace=True)
df.drop(df.query("group == 'control' and landing_page == 'new_page'").index, inplace=True)
df2 = df
df2.head()
```

Out[8]:

| | 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 [9]:

```
df2.to_csv('ab_data_2.csv')
df2 = pd.read_csv('ab_data_2.csv')
```

In [10]:

```
# 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[10]:

```
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 [11]:

```
df2.user_id.nunique()
```

Out[11]:

```
290584
```

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

In [12]:

```
df2[df2['user_id'].duplicated()]['user_id']
```

Out[12]:

```
2862    773192
Name: user_id, dtype: int64
```

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

In [13]:

```
df2[df2.duplicated(['user_id'], keep=False)]
```

Out[13]:

| | Unnamed: 0 | user_id | timestamp | group | landing_page | converted |
|---|---|---|---|---|---|---|
| **1876** | 1899 | 773192 | 2017-01-09 05:37:58.781806 | treatment | new_page | 0 |
| **2862** | 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 [14]:

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 290584 entries, 0 to 290584
Data columns (total 6 columns):
 #   Column        Non-Null Count   Dtype 
---  ------        --------------   ----- 
 0   Unnamed: 0    290584 non-null  int64 
 1   user_id       290584 non-null  int64 
 2   timestamp     290584 non-null  object
 3   group         290584 non-null  object
 4   landing_page  290584 non-null  object
 5   converted     290584 non-null  int64 
dtypes: int64(3), object(3)
memory usage: 15.5+ MB
```

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 [15]: df2['converted'].mean()
```

```
Out[15]: 0.11959708724499628
```

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

```
In [16]: df2.query('group =="control"').converted.mean()
```

```
Out[16]: 0.1203863045004612
```

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

```
In [17]: df2.query('group =="treatment"').converted.mean()
```

```
Out[17]: 0.11880806551510564
```

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

```
In [18]: (df2['landing_page'] == 'new_page').mean()
```

```
Out[18]: 0.5000619442226688
```

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.

Based on the answer to question 4a/b, the percentage of conversion rates is 12.04% in the control group and 11.88% in the treatment group. Since there were a large number of users for the study, there is not much difference. It should also be noted that the probability that a person has received the new page is 0.5 There is therefore no evidence that a page leads to more conversions, as the probability of conversions is almost the same.

## 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.

1) Null hypothesis: $pnew$ - $pold$ <= 0 2) Alternative hyporthesis: $pnew$ - $pold$ > 0

`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 [19]:
```
p_new = df2.converted.mean()
p_new
```

Out[19]:
```
0.11959708724499628
```

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

In [20]:
```
p_old = df2.converted.mean()
p_old
```

Out[20]:
```
0.11959708724499628
```

c. What is $n_{new}$?

In [21]:
```
n_new = df2.query('landing_page == "new_page"')['converted'].count()
n_new
```

Out[21]:
```
145310
```

d. What is $n_{old}$?

In [22]:
```
n_old = df2.query('landing_page == "old_page"')['converted'].count()
n_old
```

Out[22]:
```
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 [23]:
```
new_page_converted = np.random.choice([0,1], size = n_new, p=[(1-p_new), p_new])
new_page_converted.mean()
```

Out[23]:
```
0.11869107425504094
```

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 [24]:
```
old_page_converted = np.random.choice([0,1], size = n_old, p=[(1-p_old), p_old])
old_page_converted.mean()
```

Out[24]:
```
0.1194088412241695
```

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

In [25]:
```
new_page_converted.mean() - old_page_converted.mean()
```

Out[25]:
```
-0.0007177669691285637
```

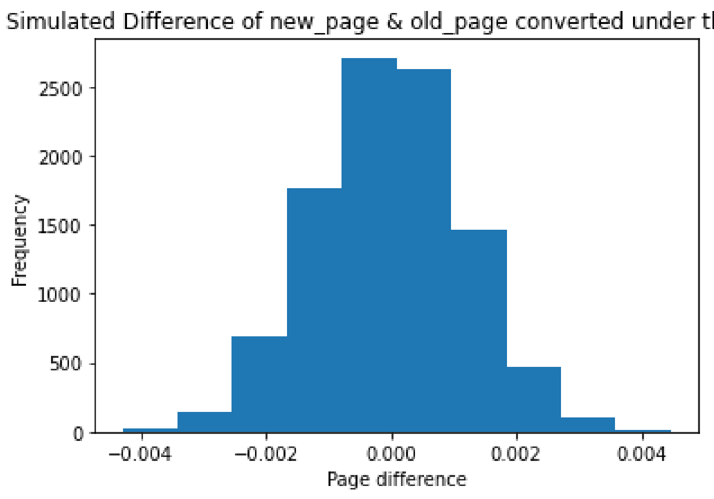
h. Simulate 10,000 $p_{new}$ - $p_{old}$ values using this same process similarly to the one you calculated in parts **a. through g.** above. Store all 10,000 values in a numpy array called **p_diffs**.

In [26]:
```
p_diffs = []
for i in range(10000):
    new_page_converted = np.random.choice([0,1], size=n_new, p=[(1-p_new), p_new]).mean()
    old_page_converted = np.random.choice([0,1], size=n_old, p=[(1-p_old), p_old]).mean()
    dif = new_page_converted - old_page_converted
    p_diffs.append(dif)
```

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 [27]:
```
p_diffs=np.array(p_diffs)
#histogram of p_diff
plt.hist(p_diffs)
plt.title('Simulated Difference of new_page & old_page converted under the Null')
plt.xlabel('Page difference')
plt.ylabel('Frequency')
```

Out[27]:
```
Text(0, 0.5, 'Frequency')
```



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

In [28]:
```
new_page_converted_act = df.query('landing_page == "new_page"')['converted'].mean()
old_page_converted_act = df.query('landing_page == "old_page"')['converted'].mean()
diff_act = new_page_converted_act - old_page_converted_act
(p_diffs > diff_act).mean()
```

Out[28]:
```
0.9037
```

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?

The p-value was calculated. This is the probability that our statistics will be observed as to whether or not the null hypothesis is true

A high p-value (in this case: 0.9) also means that the statistics are more likely to come from our null hypothesis. Therefore, there is no statistical evidence that rejects the null hypothesis. In this case: that old pages are the same or slightly better than the new 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 [29]:
```
import statsmodels.api as sm

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

Out[29]:
```
(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 [30]:
```
z_score, p_value = sm.stats.proportions_ztest([convert_old, convert_new], [n_old, n_new], alter
z_score, p_value
```

Out[30]:
```
(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 [31]:
```
from scipy.stats import norm
#how significant our z_score is
norm.cdf(z_score)
#critical value of 95% confidence
norm.ppf(1-(0.05))
```

Out[31]:
```
1.6448536269514722
```

We found that the Z score of 1.31 is below the critical value of 1.64. So we accept the null hypothesis. The results in parts j and k are consistent with this calculation. There is not enough evidence that the new pages are better or worse than the old pages.

## 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?

using 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 [46]:
```
df3 = df2
df3['intercept'] = 1
df3['ab_page'] = pd.get_dummies(df3['group'])['treatment']
df3.head()
```

Out[46]:

|   | Unnamed: 0 | user_id | timestamp | group | landing_page | converted | intercept | ab_page |
|---|---|---|---|---|---|---|---|---|
| **0** | 0 | 851104 | 2017-01-21 22:11:48.556739 | control | old_page | 0 | 1 | 0 |
| **1** | 1 | 804228 | 2017-01-12 08:01:45.159739 | control | old_page | 0 | 1 | 0 |

| | Unnamed: 0 | user_id | timestamp | group | landing_page | converted | intercept | ab_page |
|---|---|---|---|---|---|---|---|---|
| **2** | 2 | 661590 | 2017-01-11 16:55:06.154213 | treatment | new_page | 0 | 1 | 1 |
| **3** | 3 | 853541 | 2017-01-08 18:28:03.143765 | treatment | new_page | 0 | 1 | 1 |
| **4** | 4 | 864975 | 2017-01-21 01:52:26.210827 | control | old_page | 1 | 1 | 0 |

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 [47]:
```python
import statsmodels.api as sm
model = sm.Logit(df3['converted'], df3[['intercept', 'ab_page']])
results = model.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 [48]:
```python
results.summary()
```

Out[48]:

Logit Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | converted | **No. Observations:** | 290584 |
| **Model:** | Logit | **Df Residuals:** | 290582 |
| **Method:** | MLE | **Df Model:** | 1 |
| **Date:** | Tue, 14 Mar 2023 | **Pseudo R-squ.:** | 8.077e-06 |
| **Time:** | 16:45:47 | **Log-Likelihood:** | -1.0639e+05 |
| **converged:** | True | **LL-Null:** | -1.0639e+05 |
| **Covariance Type:** | nonrobust | **LLR p-value:** | 0.1899 |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **intercept** | -1.9888 | 0.008 | -246.669 | 0.000 | -2.005 | -1.973 |
| **ab_page** | -0.0150 | 0.011 | -1.311 | 0.190 | -0.037 | 0.007 |

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**?

The Logistic Regression (Part III) is a two-tailed test, which is why we get a different result (0.19) than the previous one-tailed test. The alternative hypothesis model (one-tailed test) assumed that the probability that old and new users would convert pages was equally high.

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 would be good to include factors (age, gender and nationality) that influence user behavior. But before you include them in the regression model, you should be sure that they are relevant to the analysis. Furthermore, one should note the Simpson paradox, which occurs. This phenomenon occurs when confounding variables are not considered in statistical analysis.

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 [49]:
```
countries_df = pd.read_csv('./countries.csv')
countries_df.head()
```

Out[49]:

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

In [50]:
```
countries_df.country.value_counts()
```

Out[50]:
```
US    203619
UK     72466
CA     14499
Name: country, dtype: int64
```

In [51]:
```
### Create the necessary dummy variables
df_new = countries_df.set_index('user_id').join(df2.set_index('user_id'), how='inner')
df_new.head()
```

Out[51]:

| | country | Unnamed: 0 | timestamp | group | landing_page | converted | intercept | ab_page |
|---|---|---|---|---|---|---|---|---|
| **user_id** | | | | | | | | |
| **834778** | UK | 143206 | 2017-01-14 23:08:43.304998 | control | old_page | 0 | 1 | 0 |
| **928468** | US | 157345 | 2017-01-23 14:44:16.387854 | treatment | new_page | 0 | 1 | 1 |
| **822059** | UK | 257177 | 2017-01-16 14:04:14.719771 | treatment | new_page | 1 | 1 | 1 |
| **711597** | UK | 48778 | 2017-01-22 03:14:24.763511 | control | old_page | 0 | 1 | 0 |
| **710616** | UK | 106686 | 2017-01-16 13:14:44.000513 | treatment | new_page | 0 | 1 | 1 |

In [54]:
```
df_new[['US', 'UK', 'CA']] = pd.get_dummies(df_new['country'])
df_new.drop(['country'], axis=1, inplace=True)
df_new.head()
```

Out[54]:

| | Unnamed: 0 | timestamp | group | landing_page | converted | intercept | ab_page | US | UK | CA |
|---|---|---|---|---|---|---|---|---|---|---|
| **user_id** | | | | | | | | | | |
| **834778** | 143206 | 2017-01-14 23:08:43.304998 | control | old_page | 0 | 1 | 0 | 0 | 1 | 0 |
| **928468** | 157345 | 2017-01-23 14:44:16.387854 | treatment | new_page | 0 | 1 | 1 | 0 | 0 | 1 |
| **822059** | 257177 | 2017-01-16 14:04:14.719771 | treatment | new_page | 1 | 1 | 1 | 0 | 1 | 0 |

| | Unnamed: 0 | timestamp | group | landing_page | converted | intercept | ab_page | US | UK | CA |
|---|---|---|---|---|---|---|---|---|---|---|
| **user_id** | | | | | | | | | | |
| **711597** | 48778 | 2017-01-22 03:14:24.763511 | control | old_page | 0 | 1 | 0 | 0 | 1 | 0 |
| **710616** | 106686 | 2017-01-16 13:14:44.000513 | treatment | new_page | 0 | 1 | 1 | 0 | 1 | 0 |

In [56]:

```
df_new['US_page'] = df_new['US']*df_new['ab_page']
df_new['UK_page'] = df_new['UK']*df_new['ab_page']
df_new.head()
```

Out[56]:

| | Unnamed: 0 | timestamp | group | landing_page | converted | intercept | ab_page | US | UK | CA | US_pag |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **user_id** | | | | | | | | | | | |
| **834778** | 143206 | 2017-01-14 23:08:43.304998 | control | old_page | 0 | 1 | 0 | 0 | 1 | 0 | |
| **928468** | 157345 | 2017-01-23 14:44:16.387854 | treatment | new_page | 0 | 1 | 1 | 0 | 0 | 1 | |
| **822059** | 257177 | 2017-01-16 14:04:14.719771 | treatment | new_page | 1 | 1 | 1 | 0 | 1 | 0 | |
| **711597** | 48778 | 2017-01-22 03:14:24.763511 | control | old_page | 0 | 1 | 0 | 0 | 1 | 0 | |
| **710616** | 106686 | 2017-01-16 13:14:44.000513 | treatment | new_page | 0 | 1 | 1 | 0 | 1 | 0 | |

In [57]:

```
model_2 = sm.Logit(df_new['converted'], df_new[['intercept', 'ab_page', 'UK', 'US']])
results_2 = model_2.fit()
```

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

In [58]:

```
results_2.summary()
```

Out[58]:

Logit Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | converted | **No. Observations:** | 290584 |
| **Model:** | Logit | **Df Residuals:** | 290580 |
| **Method:** | MLE | **Df Model:** | 3 |
| **Date:** | Tue, 14 Mar 2023 | **Pseudo R-squ.:** | 2.323e-05 |
| **Time:** | 16:51:52 | **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 |
| **UK** | 0.0099 | 0.013 | 0.743 | 0.457 | -0.016 | 0.036 |
| **US** | -0.0408 | 0.027 | -1.516 | 0.130 | -0.093 | 0.012 |

Answer: The P-value for the two countries is above 0.05, so the country variable has no influence on the conversion rate. The null hypothesis is also not rejected in this analysis.

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 [60]:
```
df_new.drop(['CA', 'US', 'UK'], axis = 1, inplace=True)
df_new.head()
```

Out[60]:

| | Unnamed: 0 | timestamp | group | landing_page | converted | intercept | ab_page | US_page | UK_page |
|---|---|---|---|---|---|---|---|---|---|
| **user_id** | | | | | | | | | |
| **834778** | 143206 | 2017-01-14 23:08:43.304998 | control | old_page | 0 | 1 | 0 | 0 | 0 |
| **928468** | 157345 | 2017-01-23 14:44:16.387854 | treatment | new_page | 0 | 1 | 1 | 0 | 0 |
| **822059** | 257177 | 2017-01-16 14:04:14.719771 | treatment | new_page | 1 | 1 | 1 | 0 | 1 |
| **711597** | 48778 | 2017-01-22 03:14:24.763511 | control | old_page | 0 | 1 | 0 | 0 | 0 |
| **710616** | 106686 | 2017-01-16 13:14:44.000513 | treatment | new_page | 0 | 1 | 1 | 0 | 1 |

In [61]:
```
model_3 = sm.Logit(df_new['converted'], df_new[['intercept','ab_page','US_page', 'UK_page']])
result_3 = model_3.fit()
result_3.summary()
```

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

Out[61]:

Logit Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | converted | **No. Observations:** | 290584 |
| **Model:** | Logit | **Df Residuals:** | 290580 |
| **Method:** | MLE | **Df Model:** | 3 |
| **Date:** | Tue, 14 Mar 2023 | **Pseudo R-squ.:** | 3.351e-05 |
| **Time:** | 16:56:01 | **Log-Likelihood:** | -1.0639e+05 |
| **converged:** | True | **LL-Null:** | -1.0639e+05 |
| **Covariance Type:** | nonrobust | **LLR p-value:** | 0.06785 |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **intercept** | -1.9888 | 0.008 | -246.669 | 0.000 | -2.005 | -1.973 |
| **ab_page** | -0.0183 | 0.013 | -1.449 | 0.147 | -0.043 | 0.006 |
| **US_page** | -0.0644 | 0.038 | -1.679 | 0.093 | -0.140 | 0.011 |
| **UK_page** | 0.0257 | 0.019 | 1.363 | 0.173 | -0.011 | 0.063 |

In [63]:
```
np.exp(result_3.params)
```

Out[63]:
```
intercept    0.136863
ab_page      0.981901
US_page      0.937618
```

```
UK_page      1.025986
dtype: float64
```

In [ ]:
```
Review including country variables
Even after adding the country variable, it is not clear that the new page does not convert more

Conclusion review
All analyses show that the new pages no longer convert users. For this reason, our recommendati
```

## Conclusions

Congratulations on completing the project!

### Gather Submission Materials

Once you are satisfied with the status of your Notebook, you should save it in a format that will make it easy for others to read. You can use the **File -> Download as -> HTML (.html)** menu to save your notebook as an .html file. If you are working locally and get an error about "No module name", then open a terminal and try installing the missing module using `pip install <module_name>` (don't include the "<" or ">" or any words following a period in the module name).

You will submit both your original Notebook and an HTML or PDF copy of the Notebook for review. There is no need for you to include any data files with your submission. If you made reference to other websites, books, and other resources to help you in solving tasks in the project, make sure that you document them. It is recommended that you either add a "Resources" section in a Markdown cell at the end of the Notebook report, or you can include a `readme.txt` file documenting your sources.

### Submit the Project

When you're ready, click on the "Submit Project" button to go to the project submission page. You can submit your files as a .zip archive or you can link to a GitHub repository containing your project files. If you go with GitHub, note that your submission will be a snapshot of the linked repository at time of submission. It is recommended that you keep each project in a separate repository to avoid any potential confusion: if a reviewer gets multiple folders representing multiple projects, there might be confusion regarding what project is to be evaluated.

It can take us up to a week to grade the project, but in most cases it is much faster. You will get an email once your submission has been reviewed. If you are having any problems submitting your project or wish to check on the status of your submission, please email us at dataanalyst-project@udacity.com. In the meantime, you should feel free to continue on with your learning journey by beginning the next module in the program.

In [ ]: