Analyze_ab_test_results_notebook

October 20, 2019

Data Analysis Project 3 - Analyze A/B Test Results
Student Name: Tadhi Al-Ali
Submit Date: Friday, 25 October

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]:
```
#Read the dataset file
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]:
```
#Find the number of rows in the dataset
df.shape[0]
```

Out[3]: 294478

c. The number of unique users in the dataset.

In [4]:
```
#Number of unique users in the dataset
df.user_id.nunique()
```

Out[4]: 290584

d. The proportion of users converted.

In [5]:
```
#Proportion of users converted
df.converted.mean()
```

Out[5]: 0.11965919355605512

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

In [6]:
```
# Table that show the number of times where the new_page and treatment don't match.
pd.crosstab(df.group, df.landing_page, margins=True)
```

Out[6]:

landing_page	new_page	old_page	All
group			
control	1928	145274	147202
treatment	145311	1965	147276
All	147239	147239	294478

The unique values of the table are control and treatment for groups, the new_page and old_page for landing_page. So, there are 1928 where new_page was associated with control and there are 1965 where old_page was associated with treatment. Then, I figure out that the number of times where new_page and treatment don't match is 1928 and add it to 1965 so equal to 3893.

```
In [7]: # Another way to find the number of times where new_page and treatment don't match
        df.query('landing_page == "new_page" and group == "control"').count()[0] + df.query('lan
```

```
Out[7]: 3893
```

f. Do any of the rows have missing values?

```
In [8]: #Search if there any missing values
        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
```

According to the following data, there are no values missing

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.drop(df.query("group == 'treatment' and landing_page == 'old_page'").index, inplace=T

        df.drop(df.query("group == 'control' and landing_page == 'new_page'").index, inplace=Tru
```

```
In [10]: df.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
```

```
In [11]: df.to_csv('ab_edited.csv', index=False)

In [12]: df2 = pd.read_csv('ab_edited.csv')

In [13]: # Double Check all of the correct rows were removed - this should be 0
         df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].sh

Out[13]: 0

In [14]: #Check the new data frame
         df2.head()
```

Out[14]:

	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

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

```
In [15]: df2.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 290585 entries, 0 to 290584
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: 11.1+ MB
```

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

```
In [16]: #How many unique user_ids are in df2?
         df2.user_id.nunique()

Out[16]: 290584
```

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

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

Out[17]: 1

```
In [18]: df2[df2.user_id.duplicated(keep=False)].user_id
```

```
Out[18]: 1876    773192
         2862    773192
         Name: user_id, dtype: int64
```

```
In [19]: #Find the duplicate id
         df2[df2.duplicated('user_id')]
```

Out[19]:

	user_id	timestamp	group	landing_page	converted
2862	773192	2017-01-14 02:55:59.590927	treatment	new_page	0

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

```
In [20]: df2[df2.user_id.duplicated(keep=False)]
```

Out[20]:

	user_id	timestamp	group	landing_page	converted
1876	773192	2017-01-09 05:37:58.781806	treatment	new_page	0
2862	773192	2017-01-14 02:55:59.590927	treatment	new_page	0

```
In [21]: #Another way to find the row information for the repeat user_id
         df2[df2.user_id == 773192]
```

Out[21]:

	user_id	timestamp	group	landing_page	converted
1876	773192	2017-01-09 05:37:58.781806	treatment	new_page	0
2862	773192	2017-01-14 02:55:59.590927	treatment	new_page	0

```
In [22]: df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 290585 entries, 0 to 290584
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: 11.1+ MB
```

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

```
In [23]: #Remove one of the rows with a duplicate user_id
         df2.drop_duplicates('user_id', inplace=True)
```

```
In [24]: #Confirm removal of one of the lines
         df2[df2.user_id == 773192]
```

```
Out[24]:       user_id                   timestamp      group landing_page  converted
        1876    773192  2017-01-09 05:37:58.781806  treatment     new_page          0
```

```
In [25]: df2.info()
```

```
&lt;class 'pandas.core.frame.DataFrame'&gt;
Int64Index: 290584 entries, 0 to 290584
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
```

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 [26]: # The probability of an individual converting regardless of the page they receive

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

```
Out[26]: 0.11959708724499628
```

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

```
In [27]: #Probability of a user converted in control group

         df2.query('group =="control"').converted.mean()
```

```
Out[27]: 0.1203863045004612
```

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

```
In [28]: #Probability of a user converted in treatment group

         df2.query('group =="treatment"').converted.mean()
```

```
Out[28]: 0.11880806551510564
```

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

```
In [29]: #Probability of a user landing on new_page

         (df2.landing_page == "new_page").mean()
```

Out[29]: 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.

According to the above data:

Given that an individual was in the control group, the probability they have converted is 0.120386.

Given that an individual was in the treatment group, the probability they have converted is 0.118808.

There is a small difference between the probability of users converted from treatment group and from control group. So, we cannot conclude that the new treatment page leads to more conversions.

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.

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 [30]:
```
#Find the proportion of converted rate assuming p_new and p_old are equal
p_new = df2['converted'].mean()
p_new
```

Out[30]: 0.11959708724499628

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

In [31]:
```python
#Find the proportion of converted rate assuming p_new and p_old are equal
p_old = df2['converted'].mean()
p_old
```

Out[31]: 0.11959708724499628

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

In [32]:
```python
#Number of users landing on new page
n_newPage = df2.query('group == "treatment"').shape[0]
n_newPage
```

Out[32]: 145310

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

In [33]:
```python
#Number of users landing on old page
n_oldPage = df2.query('group == "control"').shape[0]
n_oldPage
```

Out[33]: 145274

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 [34]:
```python
#Draw samples from a binomial distribution
new_page_converted = np.random.binomial(n_newPage,p_new)
new_page_converted
```

Out[34]: 17539

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 [35]:
```python
#Draw samples from a binomial distribution
old_page_converted = np.random.binomial(n_oldPage,p_old)
old_page_converted
```

Out[35]: 17216

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

In [36]:
```python
p_diff = (new_page_converted/n_newPage) - (old_page_converted/n_oldPage)
p_diff
```

Out[36]: 0.002193474258553263

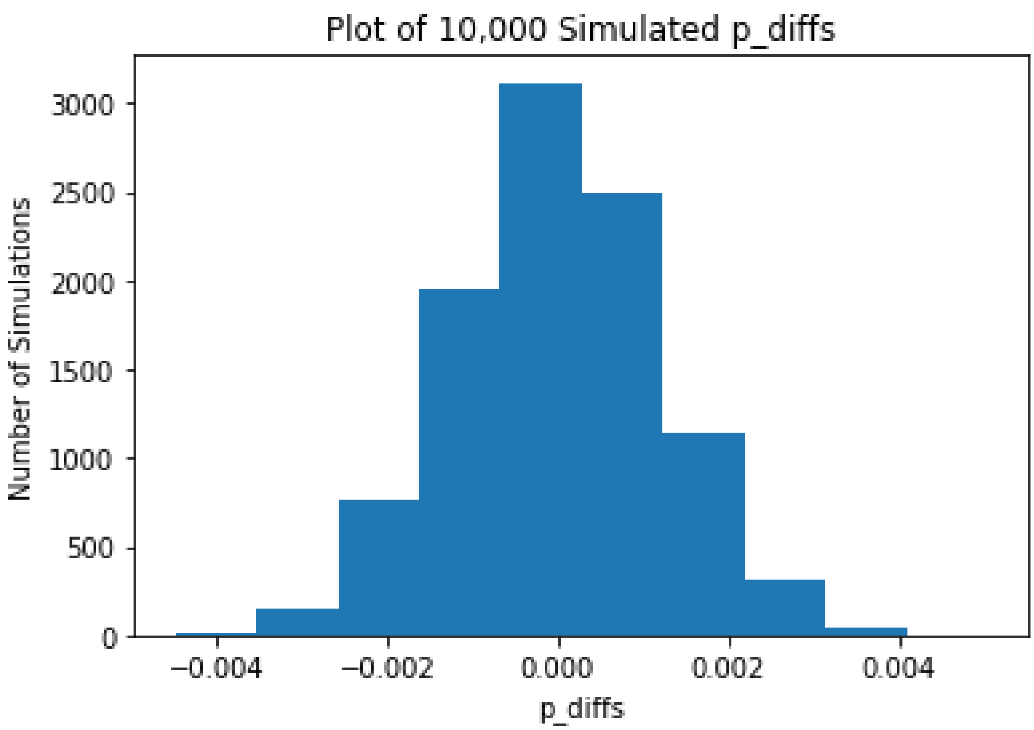
h. Create 10,000 p_{new} - p_{old} values using the same simulation process you used in parts (a) through (g) above. Store all 10,000 values in a NumPy array called **p_diffs**.

```
In [37]: #Simulate 10000 samples of the differences in conversion rates
         p_diffs = []

         for _ in range(10000):
             new_converted_simulation = np.random.binomial(n_newPage,p_new)/n_newPage
             old_converted_simulation = np.random.binomial(n_oldPage,p_old)/n_oldPage
             diff = new_converted_simulation - old_converted_simulation
             p_diffs.append(diff)
```

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 [38]: #Plot a histogram of the p_diffs.
         plt.hist(p_diffs);
         plt.ylabel('Number of Simulations')
         plt.xlabel('p_diffs')
         plt.title('Plot of 10,000 Simulated p_diffs');
```

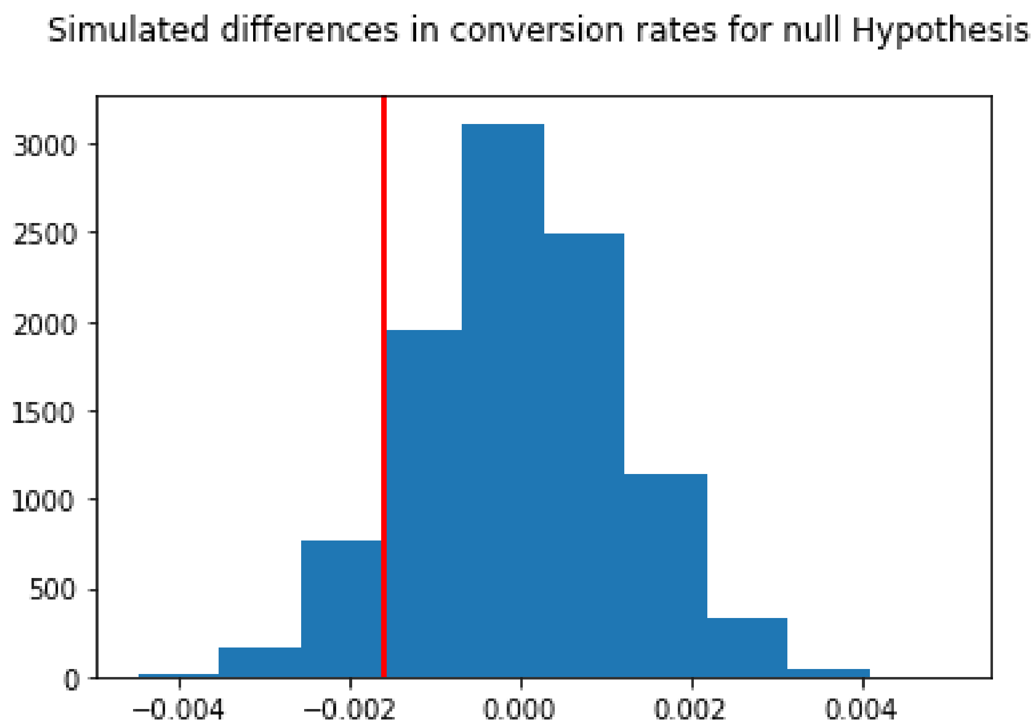


j. What proportion of the **p_diffs** are greater than the actual difference observed in **ab_data.csv**?

```
In [39]: # Calculate the actucl difference observed in ab_data
         Actual_diffs = df2.query('group == "treatment"').converted.mean() - df2.query('group ==
         Actual_diffs
```

Out[39]: -0.0015782389853555567

```
In [40]: #Plot a histogram of the p_diffs
         plt.hist(p_diffs);
         plt.title("Simulated differences in conversion rates for null Hypothesis \n");
         plt.axvline(Actual_diffs,c='r',linewidth = 2);
```




```
In [41]: #Convert to numpy array and calculate the p-value
         p_diffs = np.array(p_diffs)
         (p_diffs > Actual_diffs).mean()
```

Out[41]: 0.9014

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?

   First, the actual difference represents the difference between converted rates of new page and old page due to our data.

   Second, the p-diffs represents the simuated difference between converted rates of new page and old page due to the 10000 simulated samples.

   Thirdly, the percentage of the p-value is 90.4 which determines the probability of obtaining our observed statistic, if the null hypothesis is true.

When having a large p-value, the statistic is more likely to come from our null hypothesis.

So, there is no statistical evidence to reject the null hypothesis which states 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 [42]:
```
import statsmodels.api as sm

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

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 [43]:
```
z_score, p_value = sm.stats.proportions_ztest([convert_new, convert_old]
, [n_new, n_old], alternative='larger')
z_score, p_value
```

Out[43]:
```
(-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.**?

The negative z-score and the value of p-value suggests that we should fail to 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?

Since each row is either a conversion or no conversion, I will use 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 [44]: #Create intercept column
         df2['intercept']=1

         #Create dummies
         ab_page = ['treatment', 'control']
         df2['ab_page'] = pd.get_dummies(df2.group)['treatment']
```

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 [45]: logit = sm.Logit(df2['converted'], df2[['intercept','ab_page']])
```

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

```
In [46]: results = logit.fit()
         results.summary()

Optimization terminated successfully.
         Current function value: 0.366118
         Iterations 6


Out[46]: <class 'statsmodels.iolib.summary.Summary'>
         """
                                    Logit Regression Results
         ==============================================================================
         Dep. Variable:              converted   No. Observations:               290584
         Model:                          Logit   Df Residuals:                   290582
         Method:                           MLE   Df Model:                            1
         Date:                Sun, 20 Oct 2019   Pseudo R-squ.:               8.077e-06
         Time:                        13:39:52   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 **Part II**?

The p-value associated with ab_page column is 0.19 which is lower than the p-value calculated using the z-score function.

The p-value I obtained in the previous question corresponds to a one-tailed test while the test implied by the regression model is a two tailed test.

This is because a high p-value for ab_page in the regression model means that there is no correlation between the landing page type and the conversion rate. This implies that p_new is equal to p_old. which is the null hypothesis of a two tailed test:

Ho: p_new = p_old

H1: p_new!= p_old

The z-test in part II on the other hand is one-tailed since it has inequality signs in the hypotheses.

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?

Considering other factors is a good idea as these factors may contribute to the significance of our test results and leads to more accurate decisions. For intance, introducing the timestamp metric to determine in which part of the day the individuals converted the most.

One of the disadvantages of adding additional terms into the regression model is Simpson's paradox where the combined impact of different variables disappears or reverses when these variables are combined, but appears where these variables are tested individually.

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_df = pd.read_csv('./countries.csv')
df_new = countries_df.set_index('user_id').join(df2.set_index('user_id'), how='inner')
```

In [48]:
```
df_new['country'].unique()
```

Out[48]:
```
array(['UK', 'US', 'CA'], dtype=object)
```

In [49]:
```
df_new['country'].value_counts()
```

Out[49]:
```
US    203619
UK     72466
CA     14499
Name: country, dtype: int64
```

```
In [50]: df_new.head()
```

```
Out[50]:         country                   timestamp      group landing_page  \
        user_id
        834778       UK  2017-01-14 23:08:43.304998    control     old_page
        928468       US  2017-01-23 14:44:16.387854  treatment     new_page
        822059       UK  2017-01-16 14:04:14.719771  treatment     new_page
        711597       UK  2017-01-22 03:14:24.763511    control     old_page
        710616       UK  2017-01-16 13:14:44.000513  treatment     new_page

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

```
In [51]: # Create the necessary dummy variables
         df_new[['CA','UK','US']] = pd.get_dummies(df_new['country'])
         df_new.head()
```

```
Out[51]:         country                   timestamp      group landing_page  \
        user_id
        834778       UK  2017-01-14 23:08:43.304998    control     old_page
        928468       US  2017-01-23 14:44:16.387854  treatment     new_page
        822059       UK  2017-01-16 14:04:14.719771  treatment     new_page
        711597       UK  2017-01-22 03:14:24.763511    control     old_page
        710616       UK  2017-01-16 13:14:44.000513  treatment     new_page

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

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

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

Out[52]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              converted   No. Observations:               290584
Model:                          Logit   Df Residuals:                   290581
Method:                           MLE   Df Model:                            2
Date:                Sun, 20 Oct 2019   Pseudo R-squ.:               1.521e-05
Time:                        13:39:54   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
CA            -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
==============================================================================
"""
```

In [53]:
```python
#check for an interaction
df_new['CA_page'] = df_new['CA'] * df_new['ab_page']
df_new['UK_page'] = df_new['UK'] * df_new['ab_page']
df_new.head()
```

Out[53]:
```
        country                   timestamp      group landing_page  \
user_id
834778       UK  2017-01-14 23:08:43.304998    control     old_page
928468       US  2017-01-23 14:44:16.387854  treatment     new_page
822059       UK  2017-01-16 14:04:14.719771  treatment     new_page
711597       UK  2017-01-22 03:14:24.763511    control     old_page
710616       UK  2017-01-16 13:14:44.000513  treatment     new_page

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

In [54]:
```python
log_mod = sm.Logit(df_new['converted'], df_new[['intercept', 'ab_page', 'CA', 'UK', 'CA
result = log_mod.fit()
result.summary()
```

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

Out[54]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              converted   No. Observations:               290584
Model:                          Logit   Df Residuals:                   290578
Method:                           MLE   Df Model:                            5
Date:                Sun, 20 Oct 2019   Pseudo R-squ.:               3.482e-05
Time:                        13:39:55   Log-Likelihood:            -1.0639e+05
converged:                       True   LL-Null:                   -1.0639e+05
Covariance Type:            nonrobust   LLR p-value:                    0.1920
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -1.9865      0.010   -206.344      0.000      -2.005      -1.968
ab_page       -0.0206      0.014     -1.505      0.132      -0.047       0.006
CA            -0.0175      0.038     -0.465      0.642      -0.091       0.056
UK            -0.0057      0.019     -0.306      0.760      -0.043       0.031
CA_page       -0.0469      0.054     -0.872      0.383      -0.152       0.059
UK_page        0.0314      0.027      1.181      0.238      -0.021       0.084
==============================================================================
"""
```

1 Conclusions

From the regression above, I figure out that the p-value is higher in UK than in Canada, which means that users in the UK are more likely to convert, but still not enough evidence to reject the null hypothesis.

will fail to reject the null and conclude that there is not sufficient evidence to suggest that there is an interaction between country and page received that will predict whether a user converts or not.

Also,there is no sufficient evidence to suggest that the new page results in more conversions than the old page.

In [55]:
```
from subprocess import call
call(['python', '-m', 'nbconvert', 'Analyze_ab_test_results_notebook.ipynb'])
```

Out[55]: 0

In [ ]: