Analyze_ab_test_results_notebook

February 16, 2020

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

- Section **??**
- Section **??**
- Section **??**
- Section **??**

Introduction

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

For this project, you will be working to understand the results of an A/B test run by an e-commerce website. Your goal is to work through this notebook to help the company understand if they should implement the new page, keep the old page, or perhaps run the experiment longer to make their decision.

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

Part I - Probability

To get started, let's import our libraries.

```
In [1]: import pandas as pd
        import numpy as np
        import random
        import matplotlib.pyplot as plt
        %matplotlib inline
        #We are setting the seed to assure you get the same answers on quizzes as we set up
        random.seed(42)
```

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

a. Read in the dataset and take a look at the top few rows here:

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

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

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

```
In [3]: df.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.user_id.nunique()
```

```
Out[5]: 290584
```

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

```
In [6]: df.query("(group != 'treatment' and landing_page == 'new_page') or (group == 'treatment'
```

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

f. Do any of the rows have missing values?

```
In [7]: df.shape[0] - df.dropna().shape[0]
```

```
Out[7]: 0
```

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

a. Now use the answer to the quiz to create a new dataset that meets the specifications from the quiz. Store your new dataframe in **df2**.

```
In [8]: df2 = df.query("(group == 'control' and landing_page == 'old_page') or (group == 'treatm
```

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

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

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

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

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

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

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

```
In [11]: print(df2[df2['user_id'].duplicated()]['user_id'].to_string(index=False))
```

```
773192
```

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

```
In [12]: df2[df2.duplicated(['user_id'])]
```

```
Out[12]:       user_id                   timestamp      group landing_page  converted
         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 [13]: df2.drop_duplicates(['user_id'], keep='first',inplace=True)
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
  """Entry point for launching an IPython kernel.
```

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

```
Out[14]: 0.11959708724499628
```

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

```
In [15]: df2[df2['group'] == 'control']['converted'].mean()
```

```
Out[15]: 0.1203863045004612
```

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

```
In [16]: df2[df2['group'] == 'treatment']['converted'].mean()
```

```
Out[16]: 0.11880806551510564
```

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

```
In [17]: len(df2[df2['landing_page'] == 'new_page'].index)/len(df2.index)
```

```
Out[17]: 0.5000619442226688
```

```
In [54]: obs_diff = (df2.query('group == "control" & converted == 1').shape[0] / df2.query('grou
         df2.query('group == "treatment" & converted == 1').shape[0] / df2.query('group == "trea

         obs_diff*=-1
         obs_diff
```

```
Out[54]: -0.0015782389853555567
```

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.

The control group's conversion rate is 0.1204. The treatment group's conversion rate is 0.1188. The difference in the conversion rate between the control and treatment group is only .15 percent. It does not indicate that there is sufficient evidence to suggest 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.

Put your answer here.

`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 [34]: p_new = df2.query('converted == ' + str(1) ).shape[0] / df2.shape[0]
         p_new

Out[34]: 0.11959708724499628

In [35]: p_new = df2.converted.mean()
         p_new

Out[35]: 0.11959708724499628
```

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

```
In [36]: p_old = df2.query('converted == ' + str(1) ).shape[0] / df2.shape[0]
         p_old

Out[36]: 0.11959708724499628
```

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

```
In [37]: n_new = df2.query('landing_page == "new_page"').shape[0]
         n_new

Out[37]: 145310
```

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

```
In [38]: n_old = df2.query('landing_page == "old_page"').shape[0]
         n_old

Out[38]: 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 [39]: new_page_converted = np.random.choice([0,1], n_new, p = (p_new, 1 - p_new))
         new_page_converted

Out[39]: array([0, 1, 1, ..., 1, 1, 0])
```

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

```
Out[40]: array([1, 1, 1, ..., 1, 1, 1])
```

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

```
In [42]: new_page_converted.mean() - old_page_converted.mean()
```

```
Out[42]: 0.0013581361081702603
```

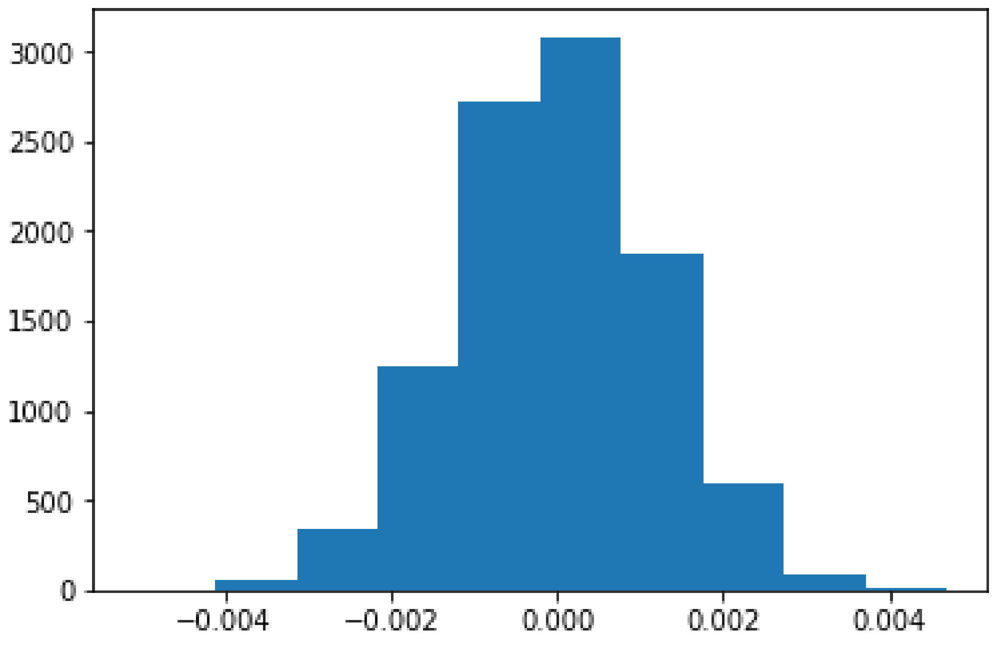
h. Create 10,000 p_{new} - p_{old} values using the same simulation process you used in parts (a) through (g) above. Store all 10,000 values in a NumPy array called **p_diffs**.

```
In [43]: import timeit
         start = timeit.default_timer()

         # Create a sampling distribution of the difference in converts
         # with bootstrapping
         diffs = []
         size = df.shape[0]
         for _ in range(10000):
             b_samp = df.sample(size, replace=True)
             new_page_converted = np.random.choice([0,1], n_new, p = (p_new, 1 - p_new))
             old_page_converted = np.random.choice([0,1], n_old, p = (p_old, 1 - p_old))
             diffs.append(new_page_converted.mean() - old_page_converted.mean())

         #Compute python running time.
         stop = timeit.default_timer()
         print (stop - start)
```

```
758.048768402
```

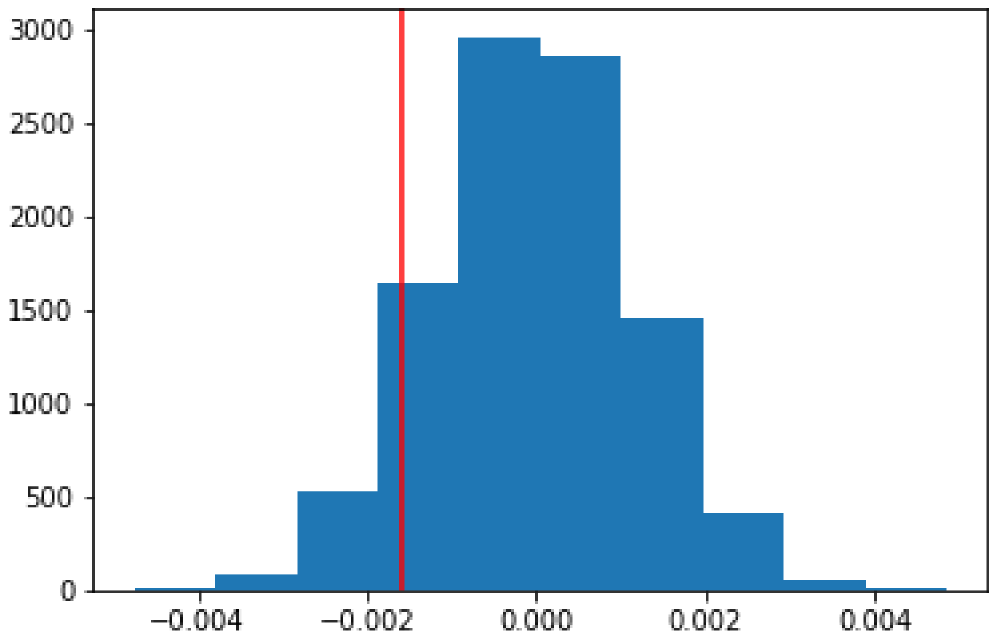
i. Plot a histogram of the **p_diffs**. Does this plot look like what you expected? Use the matching problem in the classroom to assure you fully understand what was computed here.

```
In [44]: # Convert to numpy array
         diffs = np.array(diffs)
         # Plot sampling distribution
         plt.hist(diffs);
```



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

```
In [55]: # Here are 10000 draws from the sampling distribution under the null
         null_vals = np.random.normal(0, np.std(diffs), 10000)

         # Plot observed statistic with the null distibution
         plt.hist(null_vals);
         plt.axvline(obs_diff, c='red')

Out[55]: <matplotlib.lines.Line2D at 0x7f46fc6f6400>
```



```
In [56]: #Compute proportion of the diffs are greater than the actual difference
         p_diffs = (null_vals > obs_diff).mean()
         p_diffs

Out[56]: 0.90480000000000005
```

With the null simulated distribution of the histogram a normalized set of values, the mean of the distribution falls right at zero while the actual conversion rate difference from the dataset is near the null distribution mean, implying that there is a lack of statistical departure from the null hypothes

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?

This value is called the p-value, the measurement of the probability of observing a statistic, in our case the conversion rate, or one more extreme in favor of the alternative, if the null hypothesis is true. Since the p-value is higher than the Type I error rate of 5%, we fail to reject the null hypothesis, and that the treatment page does not have higher conversion rates than the control page on a statistically significant basis. If the p-value were under 0.05, it would indicate a very low probability of, assuming the null hypothesis were true, finding a value equal to or significantly greater or lesser than obsdiff.

- Briefly, it appears that the p-value is above 0.05, which means we do not have evidence to reject the null hypothesis ($Pnew = P_{old}$).

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 [58]: 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('landing_page == "old_page"').shape[0]
         n_new = df2.query('landing_page == "new_page"').shape[0]

         convert_old, convert_new, n_old, n_new
```

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

```
Out[58]: (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 [59]: z_score, p_value = sm.stats.proportions_ztest(np.array([convert_new, convert_old]),np.a
         z_score, p_value
```

```
Out[59]: (-1.3109241984234394, 0.90505831275902449)
```

```
In [60]: from scipy.stats import norm

         print(norm.cdf(z_score)) # Tells us how significant our z-score is
         print(norm.ppf(1-(0.05/2))) # Tells us what our critical value at 95% confidence
```

```
0.094941687241
1.95996398454
```

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 z-score here is -1.31 inside our critical value of 1.959 and the p-value is still large, So it is likely that our statistic is from the null. This means the z-score and p-value agree with the findings in parts j and k that we cannot 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?

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

```
Out[61]:    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 [62]: df2[['ab_page', 'new_page']] = pd.get_dummies(df2['landing_page'])
         df2.head()
```

```
/opt/conda/lib/python3.6/site-packages/pandas/core/frame.py:3140: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
  self[k1] = value[k2]
```

```
Out[62]:    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

            ab_page  new_page
         0        0         1
         1        0         1
         2        1         0
         3        1         0
         4        0         1
```

```
In [63]: df2 = df2.drop('new_page', axis=1)
         df2.head()
```

```
Out[63]:    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

           ab_page
        0        0
        1        0
        2        1
        3        1
        4        0
```

```
In [64]: df2[['control_group', 'treatment']] = pd.get_dummies(df2['group'])
         df2.head()
```

```
Out[64]:    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

           ab_page  control_group  treatment
        0        0              1          0
        1        0              1          0
        2        1              0          1
        3        1              0          1
        4        0              1          0
```

```
In [65]: df2 = df2.drop('treatment', axis=1)
         df2['intercept'] = 1

         df2.head()
```

```
Out[65]:    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

           ab_page  control_group  intercept
        0        0              1          1
        1        0              1          1
        2        1              0          1
        3        1              0          1
        4        0              1          1
```

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 [66]: import statsmodels.api as sm

         logit_mod = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])

         results = logit_mod.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 [67]: results.summary()

Out[67]: <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, 16 Feb 2020   Pseudo R-squ.:               8.077e-06
         Time:                        17:53:29   Log-Likelihood:            -1.0639e+05
         converged:                       True   LL-Null:                   -1.0639e+05
                                                 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 for ab_page is 0.19. and this p-value is quite larger than type 1 error (.05) and this means that the landing page is not statistically significant in predicting whether the viewer converts or not.

The null and alternative hypotheses for the regression model are:

$$Null\ hypotheses: \ H_0 : p_{old} - p_{new} = 0$$

$$Alternative\ hypotheses:\ H_1: p_{old} - p_{new} =!0$$

While p-value in Part 2 the null and alternative hypotheses were:

$$Null\ hypotheses:\ H_0: p_{old} - p_{new} \geq 0$$

$$Alternative\ hypotheses:\ H_1: p_{old} - p_{new} < 0$$

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?

More factors should be incorporated into the model, since the current hypotheses only incorporated a single factor for conversion, the type of page, which can show a succinct relationship but will obviously not show insight into other factors that could influence conversion.

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 [68]: countries_df = pd.read_csv('./countries.csv')
         df_new = countries_df.set_index('user_id').join(df2.set_index('user_id'), how='inner')
         df_new.head()
```

```
Out[68]:         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  ab_page  control_group  intercept
         user_id
         834778           0        0              1          1
         928468           0        1              0          1
         822059           1        1              0          1
         711597           0        0              1          1
         710616           0        1              0          1
```

```
In [69]: df_new['country'].value_counts()
```

```
Out[69]: US    203619
         UK     72466
         CA     14499
         Name: country, dtype: int64
```

```
In [70]: ### Create the necessary dummy variables
         df_new[['CA', 'UK', 'US']] = pd.get_dummies(df_new['country'])
```

```
In [71]: logit_mod_country = sm.Logit(df_new['converted'],df_new[['intercept', 'ab_page', 'US',
```

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 [72]: new_results = logit_mod_country.fit()
         new_results.summary()
```

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

```
Out[72]: <class 'statsmodels.iolib.summary.Summary'>
         """
                                    Logit Regression Results
         ==============================================================================
         Dep. Variable:              converted   No. Observations:               290584
         Model:                          Logit   Df Residuals:                   290580
         Method:                           MLE   Df Model:                            3
         Date:                Sun, 16 Feb 2020   Pseudo R-squ.:               2.323e-05
         Time:                        17:57:10   Log-Likelihood:            -1.0639e+05
         converged:                       True   LL-Null:                   -1.0639e+05
                                                 LLR p-value:                   0.1760
         ==============================================================================
                          coef    std err          z      P>|z|      [0.025      0.975]
         ------------------------------------------------------------------------------
         intercept     -2.0300      0.027    -76.249      0.000      -2.082      -1.978
         ab_page       -0.0149      0.011     -1.307      0.191      -0.037       0.007
         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
         ==============================================================================
         """
```

```
In [77]: np.exp(new_results.params)
```

```
Out[77]: intercept    0.131332
         ab_page      0.985168
         US           1.041599
         UK           1.051944
         dtype: float64
```

```
In [78]: 1/np.exp(new_results.params)

Out[78]: intercept    7.614303
         ab_page      1.015056
         US           0.960062
         UK           0.950621
         dtype: float64
```

This regression summary shows that the country of origin and the type of page based on their p-values do not provide a statistical basis to reject the null hypothesis, based on a Type I error rate of 5%. Even with all this statistical analysis, as noted earlier, the practical significance of all this analysis does not seem required given that a .15 percent difference in the conversion rates between the control and treatment group was noted from the dataset itself, which is so minute as to not have any practical effect on deciding to switch to the new page rather than stay with the old one.
Finishing Up

> Congratulations! You have reached the end of the A/B Test Results project! You should be very proud of all you have accomplished!

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

0.3 Directions to Submit

> Before you submit your project, you need to create a .html or .pdf version of this notebook in the workspace here. To do that, run the code cell below. If it worked correctly, you should get a return code of 0, and you should see the generated .html file in the workspace directory (click on the orange Jupyter icon in the upper left).

> Alternatively, you can download this report as .html via the **File** > **Download as** submenu, and then manually upload it into the workspace directory by clicking on the orange Jupyter icon in the upper left, then using the Upload button.

> Once you've done this, you can submit your project by clicking on the "Submit Project" button in the lower right here. This will create and submit a zip file with this .ipynb doc and the .html or .pdf version you created. Congratulations!

```
In [79]: from subprocess import call
         call(['python', '-m', 'nbconvert', 'Analyze_ab_test_results_notebook.ipynb'])

Out[79]: 0

In [ ]:
```