Analyze_ab_test_results_notebook

September 17, 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

### 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.converted.mean()
```

```
Out[5]: 0.11965919355605512
```

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

```
In [6]: df_control = df[df.group == 'control']
        df_treatment = df[df.group == 'treatment']

        df_control[df_control.landing_page == 'new_page'].append(
            df_treatment[df_treatment.landing_page == 'old_page']).shape[0]
```

```
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** 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.copy()

        df2 = df2[((df2.group == 'control') & (df2.landing_page == 'old_page')) |
                  (df2.group == 'treatment') & (df2.landing_page == 'new_page')]

        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]: # 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]: df2[df2.user_id.duplicated()].iloc[0, 0]
```

```
Out[11]: 773192
```

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

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

```
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(df2[df2.user_id.duplicated()].index[0], axis=0, inplace=True)
         df2.user_id.duplicated().sum()
```

```
Out[13]: 0
```

4. Use **df2** in the cells below to answer the quiz questions related to **Quiz 4** in the classroom.

a. What is the probability of an individual converting regardless of the page they receive?

```
In [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]: df2[df2.landing_page == 'new_page'].shape[0] / df2.shape[0]
```

```
Out[17]: 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.

I think, hypothesis test need to be done to conclude whether there exists sufficient evidence that the new treatment page leads to more conversions. Control group conversion is slightly higher than treatment group conversion (about 0.0016), and probability that an individual received the new page is about 50%. So in the next Part II, this hypothesis will be tested.

### 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 **conversion rate** for p_{new} under the null?

```
In [18]: p_new = df2.converted.mean()
         p_new
```

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

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

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

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

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

```
In [20]: n_new = df2[df2.group == 'treatment'].shape[0]
         n_new
```

```
Out[20]: 145310
```

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

```
In [21]: n_old = df2[df2.group == 'control'].shape[0]
         n_old
```

```
Out[21]: 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 [22]: new_page_converted = np.random.binomial(n=1, p=p_new, size=n_new)
         new_page_converted[:10]
```

```
Out[22]: array([0, 0, 1, 0, 0, 0, 0, 0, 0, 0])
```

```
In [23]: new_page_converted.mean(), new_page_converted.std()
```

```
Out[23]: (0.11934484894363774, 0.32419385554673619)
```

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 [24]: old_page_converted = np.random.binomial(n=1, p=p_old, size=n_old)
         old_page_converted[:10]
```

```
Out[24]: array([0, 0, 0, 0, 1, 1, 0, 0, 0, 0])
```

```
In [25]: old_page_converted.mean(), old_page_converted.std()
```

```
Out[25]: (0.11974613488993213, 0.32466443918121718)
```

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

```
In [26]: obs_diff = new_page_converted.mean() - old_page_converted.mean()
         obs_diff
```

```
Out[26]: -0.00040128594629439129
```

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 [27]: p_diffs = []
         for _ in range(10000):
             new_page_converted = np.random.binomial(n=1, p=p_new, size=n_new)
             old_page_converted = np.random.binomial(n=1, p=p_old, size=n_old)
             diff = new_page_converted.mean() - old_page_converted.mean()
             p_diffs.append(diff)

         p_diffs = np.array(p_diffs)
         p_diffs[:10]
```

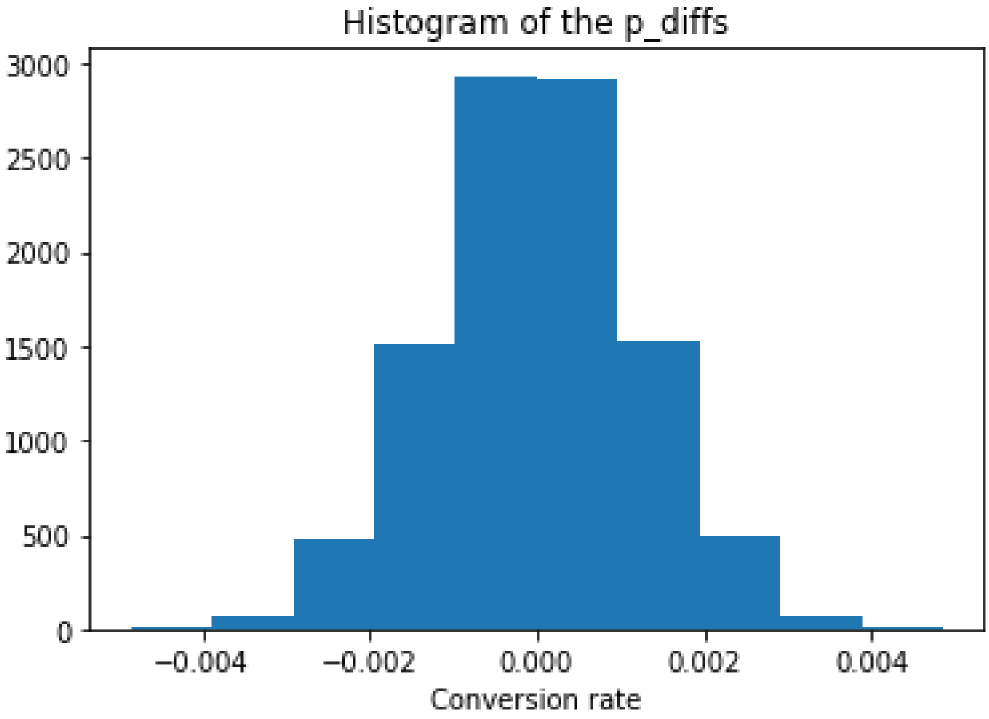
```
Out[27]: array([ -3.65484884e-05,  -2.49360029e-03,   2.52592770e-04,
                 -6.90314650e-04,   3.15035853e-03,   1.64300560e-03,
                  1.35763520e-04,  -9.86324104e-04,  -2.43126440e-04,
                  7.00024848e-04])
```

```
In [28]: p_diffs.mean(), p_diffs.std()
```

```
Out[28]: (1.6550481400102554e-06, 0.0012017929807558621)
```

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 [29]: plt.hist(p_diffs)
         plt.xlabel('Conversion rate')
         plt.title('Histogram of the p_diffs')
         plt.show()
```



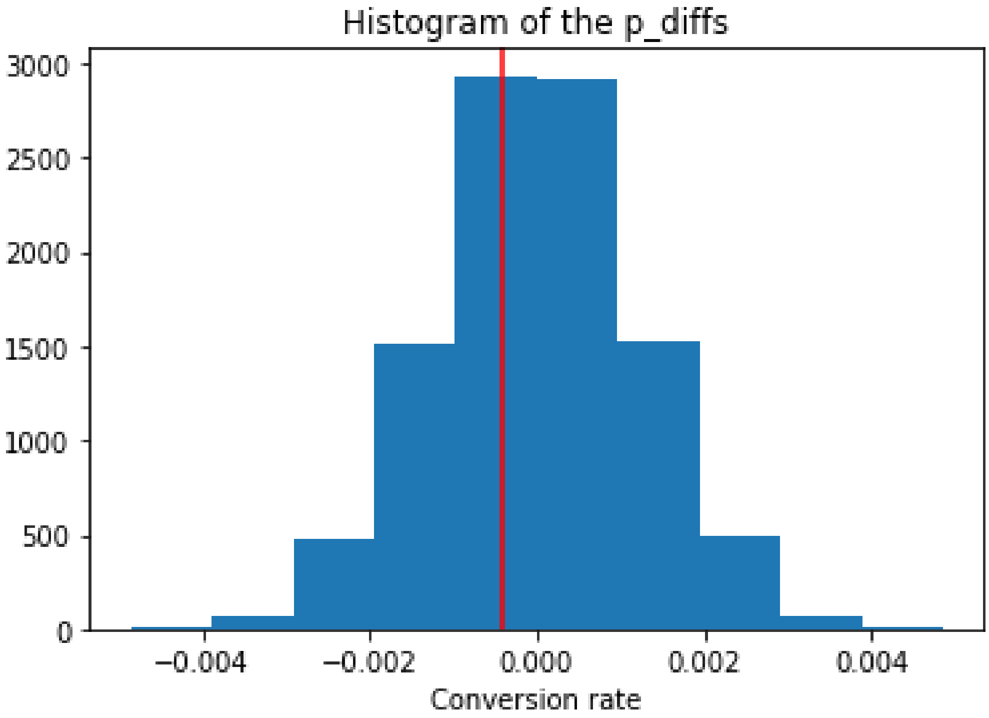
j. What proportion of the **p_diffs** are greater than the actual difference observed in **ab_data.csv**?

```
In [30]: # use original ab_data.csv data
         print((p_diffs > df_treatment.converted.mean() - df_control.converted.mean()).mean())

         # use cleaned df2 data
         print((p_diffs > obs_diff).mean())

0.8919
0.628
```

```
In [31]: plt.hist(p_diffs)
         plt.axvline(obs_diff, color='red')
         plt.xlabel('Conversion rate')
         plt.title('Histogram of the p_diffs')
         plt.show()
```



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?

Since probability, that the old page is better than the new page at a significance level of 5%, is 84.76% we can't reject the Null hypothesis. Thus, converted rate of the new page isn't higher then converted rate of old. So, company shouldn't implement the new page, and should keep the old 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 [32]: import statsmodels.api as sm

         df_control = df2[df2.group == 'control']
         df_treatment = df2[df2.group == 'treatment']

         convert_old = df_control.converted.sum()
         convert_new = df_treatment.converted.sum()
```

```
        n_old = df_control.shape[0]
        n_new = df_treatment.shape[0]
```

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

```
In [33]: convert_old, convert_new
```

```
Out[33]: (17489, 17264)
```

```
In [34]: n_old, n_new
```

```
Out[34]: (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 [35]: zstat, pval = sm.stats.proportions_ztest([convert_old,convert_new], [n_old, n_new], alt
         zstat, pval
```

```
Out[35]: (1.3109241984234394, 0.90505831275902449)
```

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.**?

At the 5% significance level, the critical value for one-tailed test statistic is 1.645. Received z-score is 1.3109 and it's less than the critical value. So, we fail to reject the Null hypothesis. This result is consistent 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 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 [36]: df2['ab_page'] = pd.get_dummies(df2.group)['treatment']
         df2['intercept'] = 1
```

```
In [37]: df2.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

           ab_page  intercept
        0        0          1
        1        0          1
        2        1          1
        3        1          1
        4        0          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 [38]: logit_model = sm.Logit(df2.converted, df2[['intercept', 'ab_page']])
         results = logit_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 [39]: results.summary2()
```

```
Out[39]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                  Results: Logit
         ==================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2020-09-17 06:25 AIC:              212780.3502
         No. Observations:   290584           BIC:              212801.5095
         Df Model:           1                Log-Likelihood:   -1.0639e+05
         Df Residuals:       290582           LL-Null:          -1.0639e+05
         Converged:          1.0000           Scale:            1.0000
         -------------------------------------------------------------------
                      Coef.   Std.Err.      z     P>|z|    [0.025   0.975]
         -------------------------------------------------------------------
         intercept   -1.9888    0.0081  -246.6690 0.0000  -2.0046  -1.9730
         ab_page     -0.0150    0.0114    -1.3109 0.1899  -0.0374   0.0074
         ==================================================================

         """
```

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**?

P-value associated with ab_page equals 0.1899. It's differ from the value found in Part II because we used one-sided test, and in statsmodels Logit, two one-sided tests are used.

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?

Among external factors, individual characteristics (e.g. age, gender, education, country of residence, etc.) could influence conversion of users. Company activities may also indirectly influence the results. For example, companies conduct promotions and campaigns.
On the one hand, collecting more data could lead to the appearance of a large number of mistakes / errors in data.
On the other hand, if data was collected correctly, adding new factors could lead to improving regression results. So the more data, the more chaances we find factors which more fully explain variance of dependent variable - whether or not an individual converts.

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 [40]: countries = pd.read_csv('countries.csv')
         print('Shape', countries.shape)
         countries.head()
```

```
Shape (290584, 2)
```

```
Out[40]:    user_id country
         0   834778      UK
         1   928468      US
         2   822059      UK
         3   711597      UK
         4   710616      UK
```

```
In [41]: df_merged = df2.merge(countries, on='user_id', how='left')
         df_merged.head()
```

```
Out[41]:    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  intercept country
        0        0          1      US
        1        0          1      US
        2        1          1      US
        3        1          1      US
        4        0          1      US
```

```
In [42]: # Check missing values in country column
         df_merged.country.isnull().sum()
```

```
Out[42]: 0
```

```
In [43]: # Unique values in country column
         df_merged.country.unique()
```

```
Out[43]: array(['US', 'CA', 'UK'], dtype=object)
```

```
In [44]: df_merged[['CA', 'UK']] = pd.get_dummies(df_merged.country)[['CA', 'UK']]
         df_merged.head()
```

```
Out[44]:    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  intercept country  CA  UK
         0        0          1      US   0   0
         1        0          1      US   0   0
         2        1          1      US   0   0
         3        1          1      US   0   0
         4        0          1      US   0   0
```

```
In [45]: logit_model_2 = sm.Logit(df_merged.converted, df_merged[['intercept', 'CA', 'UK']])
         results_2 = logit_model_2.fit()
```

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

```
In [46]: results_2.summary2()
```

```
Out[46]: <class 'statsmodels.iolib.summary2.Summary'>
"""
                          Results: Logit
==================================================================
Model:              Logit            No. Iterations:   6.0000
Dependent Variable: converted        Pseudo R-squared: 0.000
Date:               2020-09-17 06:25 AIC:              212780.8333
No. Observations:   290584           BIC:              212812.5723
Df Model:           2                Log-Likelihood:   -1.0639e+05
Df Residuals:       290581           LL-Null:          -1.0639e+05
Converged:          1.0000           Scale:            1.0000
-------------------------------------------------------------------
              Coef.  Std.Err.      z      P>|z|    [0.025   0.975]
-------------------------------------------------------------------
intercept    -1.9967    0.0068  -292.3145  0.0000  -2.0101  -1.9833
CA           -0.0408    0.0269    -1.5178  0.1291  -0.0935   0.0119
UK            0.0099    0.0133     0.7458  0.4558  -0.0161   0.0360
==================================================================

"""
```

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 [47]: df_merged['ab_page_and_CA'] = df_merged['ab_page'] * df_merged['CA']
         df_merged['ab_page_and_UK'] = df_merged['ab_page'] * df_merged['UK']

         df_merged.head()
```

```
Out[47]:    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  intercept country  CA  UK  ab_page_and_CA  ab_page_and_UK
         0        0          1      US   0   0               0               0
         1        0          1      US   0   0               0               0
         2        1          1      US   0   0               0               0
         3        1          1      US   0   0               0               0
         4        0          1      US   0   0               0               0
```

```
In [48]: # Proportions of new columns
         df_merged['ab_page_and_CA'].mean(), df_merged['ab_page_and_UK'].mean()
```

```
Out[48]: (0.025125264983619194, 0.12425322798227019)
```

```
In [49]: logit_model_3 = sm.Logit(df_merged.converted, df_merged[['intercept', 'ab_page', 'CA',
                                                                   'ab_page_and_UK']])
         results_3 = logit_model_3.fit()
```

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

```
In [50]: results_3.summary2()
```

```
Out[50]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                  Results: Logit
         ==================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2020-09-17 06:25 AIC:              212782.6602
         No. Observations:   290584           BIC:              212846.1381
         Df Model:           5                Log-Likelihood:   -1.0639e+05
         Df Residuals:       290578           LL-Null:          -1.0639e+05
         Converged:          1.0000           Scale:            1.0000
         ------------------------------------------------------------------
                           Coef.  Std.Err.     z     P>|z|   [0.025  0.975]
         ------------------------------------------------------------------
         intercept        -1.9865   0.0096 -206.3440 0.0000 -2.0053 -1.9676
         ab_page          -0.0206   0.0137   -1.5052 0.1323 -0.0473  0.0062
         CA               -0.0175   0.0377   -0.4652 0.6418 -0.0914  0.0563
         UK               -0.0057   0.0188   -0.3057 0.7598 -0.0426  0.0311
         ab_page_and_CA   -0.0469   0.0538   -0.8718 0.3833 -0.1523  0.0585
         ab_page_and_UK    0.0314   0.0266    1.1807 0.2377 -0.0207  0.0835
         ==================================================================

         """
```

Unfortunately, none of the above factors (page, country, or an interaction between page and country) didn't bring any significant results. Therefore, logistic regression didn't show better conversion of new page compared with an old page.
So, this analysis is open to future research.

## Conclusions

In this project, I was working to understand the results of an A/B test run by an e-commerce website. I would suggest the company not implement the new page immediately, and on continue to keep the old page. Perhaps, if the company has resources and time, it's possible to run the experiment longer to make their final decision later.

```
In [51]: from subprocess import call
         call(['python', '-m', 'nbconvert', 'Analyze_ab_test_results_notebook.ipynb'])
```

```
Out[51]: 0
```