Analyze_ab_test_results_notebook

March 23, 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
```

```
Out[3]: (294478, 5)
```

c. The number of unique users in the dataset.

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

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

d. The proportion of users converted.

```
In [5]: usesrs_converted = df[df['converted'] == 1]['user_id'].nunique()
        usesrs_converted
```

```
Out[5]: 35173
```

```
In [6]: p_converted = usesrs_converted/un_users
        p_converted
```

```
Out[6]: 0.12104245244060237
```

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

```
In [7]: df[((df['group'] == 'treatment') != (df['landing_page'] == 'new_page'))].shape
```

```
Out[7]: (3893, 5)
```

f. Do any of the rows have missing values?

```
In [8]: 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
```

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]: df2 = df.drop(df[((df['group'] == 'treatment') != (df['landing_page'] == 'new_page'))].i

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].sh

Out[10]: 0

In [11]: df2.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
```

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

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

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

Out[12]: 290584
```

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

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

```
Out[13]:       user_id                   timestamp      group landing_page  converted
         2893   773192  2017-01-14 02:55:59.590927  treatment     new_page          0
```

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

```
In [14]: df2[df2['user_id'] == 773192]
```

```
Out[14]:       user_id                   timestamp      group landing_page  converted
         1899   773192  2017-01-09 05:37:58.781806  treatment     new_page          0
         2893   773192  2017-01-14 02:55:59.590927  treatment     new_page          0
```

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

```
In [15]: df2.drop(index=1899, inplace=True)
```

```
In [16]: df2[df2['user_id'] == 773192]
```

```
Out[16]:       user_id                   timestamp      group landing_page  converted
         2893   773192  2017-01-14 02:55:59.590927  treatment     new_page          0
```

```
In [17]: df2.shape
```

```
Out[17]: (290584, 5)
```

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 [18]: p_converted2 = df2['converted'].mean()
         p_converted2
```

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

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

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

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

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

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

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

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

```
In [21]: p1_new = df2.query("landing_page == 'new_page'").count()[0]/df2.shape[0]
         p1_new

Out[21]: 0.50006194422266881

In [22]: # Probability that an individual received the old page (for later use in part II)
         p1_old = df2.query("landing_page == 'old_page'").count()[0]/df2.shape[0]
         p1_old

Out[22]: 0.49993805577733119

In [23]: p1_new - p1_old

Out[23]: 0.00012388844533761656
```

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.

Considering the previous results, there is a very slight difference in probabilities from which we can not conclude that the new treatment pages leads to more conversions. In fact, the probability of users converting from the old page is higher by 0.16%

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 [24]: # Assuming p_new and p_old are equal and both are equal to the converted rate in ab_dat
         p_new = df2['converted'].mean()
         p_new

Out[24]: 0.11959708724499628
```

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

```
In [25]: p_old = p_new
         p_old

Out[25]: 0.11959708724499628
```

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

```
In [26]: n_new = df2.query('group == "treatment"').count()[0]
         n_new

Out[26]: 145310
```

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

```
In [27]: n_old = df2.query('group == "control"').count()[0]
         n_old

Out[27]: 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 [28]: new_page_converted = np.random.choice([0,1], n_new, p=[p_new,(1-p_new)])
         new_page_converted

Out[28]: array([1, 1, 1, ..., 1, 1, 1])
```

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

Out[29]: array([1, 1, 1, ..., 1, 1, 1])
```

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

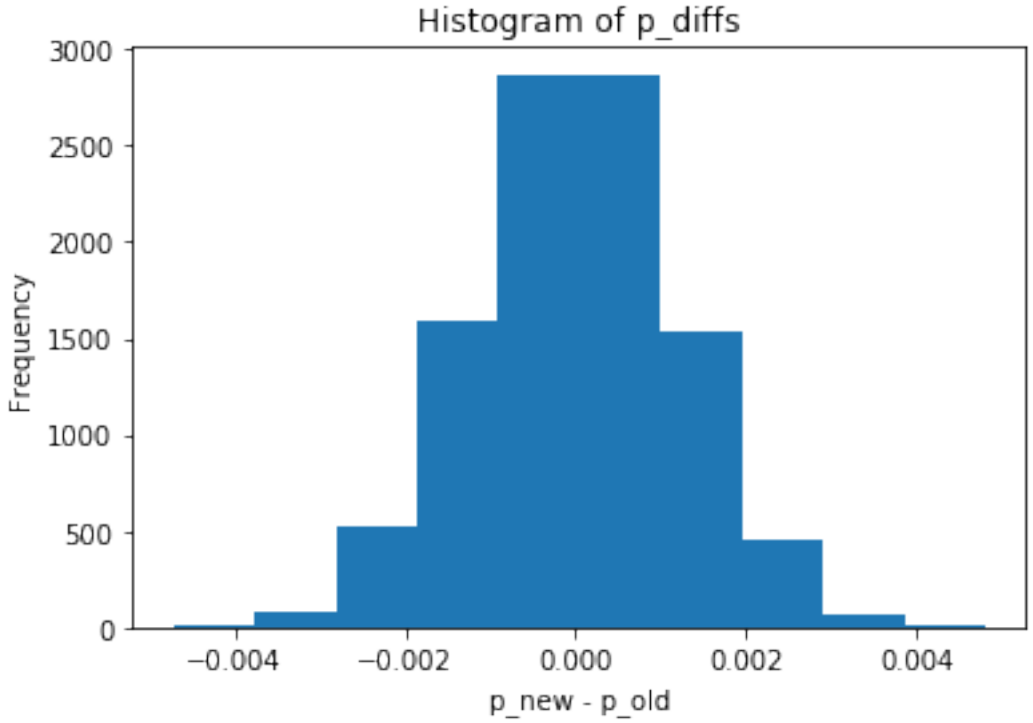
```
In [30]: new_page_converted.mean() - old_page_converted.mean()

Out[30]: -3.2309123381479843e-05
```

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 [31]: p_diffs = []
         for _ in range(10000):
             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)])
             diff = new_page_converted.mean() - old_page_converted.mean()
             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 [32]: plt.hist(p_diffs)
         plt.xlabel('p_new - p_old')
         plt.ylabel('Frequency')
         plt.title('Histogram of p_diffs');
```



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

```
In [33]: actual_diff = pt_converted - pc_converted
         actual_diff

Out[33]: -0.0015782389853555567

In [34]: # Proportion of the p_diffs are greater than the actual difference
         (p_diffs > actual_diff).mean()
```

```
Out[34]: 0.90449999999999997
```

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?

I just computed the P-value. Since P-value is high, we fail to reject the Null. If it had been less than Type I error rate of 5%, the new page would have proved to be definitely better than 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 [35]: import statsmodels.api as sm

         convert_old = df2.query('landing_page == "old_page" and converted == 1').count()[0]
         convert_old
         convert_new = df2.query('landing_page == "new_page" and converted == 1').count()[0]
         n_old = n_old
         n_new = n_new

         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[35]: (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 [36]: z_score, p_value = sm.stats.proportions_ztest([convert_old, convert_new], [n_old, n_new
         z_score, p_value # Alternative = smaller means p1<p2 in Alt hypothesis
```

```
Out[36]: (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.**?

```
In [37]: from scipy.stats import norm
         # Significance of z-score using Cumulative Density FUnction
         print(norm.cdf(z_score))

         # Critical value at 95% confidence (5% Type I error) using Percent Point Function
         print(norm.ppf(1-(0.05)))
```

```
0.905058312759
1.64485362695
```

Since the z-score is less than the critical value, we fail to reject the null, meaning that the conversion rates of the old page is more than or equal to that of the new page. This agrees 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 [38]: df2['intercept'] = 1
         df2['ab_page'] = pd.get_dummies(df['group']) ['treatment']
         df2.head()
```

Out[38]:

	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	

	intercept	ab_page
0	1	0
1	1	0
2	1	1
3	1	1
4	1	0

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 [42]: import statsmodels.api as sm
         import scipy.stats as stats

         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 [45]: # Results.summary didn't work. This is a workaround from the statsmodels documentation
         # https://www.statsmodels.org/stable/generated/statsmodels.discrete.discrete_model.Logi
         results.summary2()
```

```
Out[45]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                  Results: Logit
         =================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2020-03-23 02:34 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**?

The p-value here is 0.19. Logistic Regression helps us to predict only one of two possible outcomes; whether an indiviual will click to the wbsite or not depending on the page he views. Smaller p-values (closer to 0) suggest that these vaiables are statistically significant in realting to the response variable. Here, a p-value of 0.19 is insignificant. This is different from the null and alternative hypotheses ini part II which assume that the old page is better or produces equal conversions to those from the new page unless the new page proves to be definitely better at a Type I error rate of 5%

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?

Another factor that may influence conversion is the period of time a user gets exposed to the page design. Users may "get used" to one model and so prefer it to the other. May be

after enough time using the new page, more users will convert. However, one disadvantage of adding more factors to the regression model is muticolinearity where factors are related to,or dependent, on each other. This makes it more difficult to interpret the results.

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 = pd.read_csv('countries.csv')
         countries.head()

Out[47]:    user_id country
         0   834778      UK
         1   928468      US
         2   822059      UK
         3   711597      UK
         4   710616      UK

In [55]: countries.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 290584 entries, 0 to 290583
Data columns (total 2 columns):
user_id    290584 non-null int64
country    290584 non-null object
dtypes: int64(1), object(1)
memory usage: 4.4+ MB


In [58]: countries.country.unique()

Out[58]: array(['UK', 'US', 'CA'], dtype=object)

In [59]: joined_df = df2.join(countries.set_index('user_id'), on = 'user_id')
         joined_df.head()

Out[59]:    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

            intercept  ab_page country
```

```
0          1        0      US
1          1        0      US
2          1        1      US
3          1        1      US
4          1        0      US
```

In [73]:
```
joined_df[['CA','UK','US']]=pd.get_dummies(joined_df['country']) #dummies must be liste
joined_df.head(10)
```

Out[73]:

	user_id	timestamp	group	landing_page	converted	intercept	ab_page	country	US	UK	CA
0	851104	2017-01-21 22:11:48.556739	control	old_page	0	1	0	US	1	0	0
1	804228	2017-01-12 08:01:45.159739	control	old_page	0	1	0	US	1	0	0
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0	1	1	US	1	0	0
3	853541	2017-01-08 18:28:03.143765	treatment	new_page	0	1	1	US	1	0	0
4	864975	2017-01-21 01:52:26.210827	control	old_page	1	1	0	US	1	0	0
5	936923	2017-01-10 15:20:49.083499	control	old_page	0	1	0	US	1	0	0
6	679687	2017-01-19 03:26:46.940749	treatment	new_page	1	1	1	CA	0	0	1
7	719014	2017-01-17 01:48:29.539573	control	old_page	0	1	0	US	1	0	0
8	817355	2017-01-04 17:58:08.979471	treatment	new_page	1	1	1	UK	0	1	0
9	839785	2017-01-15 18:11:06.610965	treatment	new_page	1	1	1	CA	0	0	1

In [76]:
```
joined_df['intercept'] = 1
logit_mod2 = sm.Logit(joined_df['converted'], joined_df[['intercept', 'UK', 'CA']])
results2 = logit_mod2.fit()
results2.summary2()
```

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

Out[76]:
```
<class 'statsmodels.iolib.summary2.Summary'>
"""
                          Results: Logit
=================================================================
Model:              Logit            No. Iterations:   6.0000
Dependent Variable: converted        Pseudo R-squared: 0.000
```

```
Date:                2020-03-23 04:23 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
UK            0.0099    0.0133     0.7458  0.4558  -0.0161   0.0360
CA           -0.0408    0.0269    -1.5178  0.1291  -0.0935   0.0119
==================================================================

"""
```

P-values indicate that countries do not have an impact on conversion.

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 [77]: logit_mod3 = sm.Logit(joined_df['converted'], joined_df[['intercept', 'ab_page', 'UK',
         results3 = logit_mod3.fit()
         results3.summary2()

Optimization terminated successfully.
         Current function value: 0.366113
         Iterations 6


Out[77]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                 Results: Logit
         ==================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2020-03-23 04:24 AIC:              212781.1253
         No. Observations:   290584           BIC:              212823.4439
         Df Model:           3                Log-Likelihood:   -1.0639e+05
         Df Residuals:       290580           LL-Null:          -1.0639e+05
         Converged:          1.0000           Scale:            1.0000
         ------------------------------------------------------------------
                        Coef.   Std.Err.      z      P>|z|    [0.025   0.975]
         ------------------------------------------------------------------
         intercept    -1.9893    0.0089  -223.7628  0.0000  -2.0067  -1.9718
         ab_page      -0.0149    0.0114    -1.3069  0.1912  -0.0374   0.0075
         UK            0.0099    0.0133     0.7433  0.4573  -0.0162   0.0359
```

```
CA           -0.0408    0.0269    -1.5161   0.1295   -0.0934    0.0119
==============================================================================

"""
```

```
In [78]: np.exp(results3.params)
```

```
Out[78]: intercept    0.136795
         ab_page      0.985168
         UK           1.009932
         CA           0.960062
         dtype: float64
```

From the coefficients, we can conclude that:

1. If a user uses the new page, it is 0.985 more likely that they convert, holding all other variables constant.

2. If a user is from the UK, it is 1.02 more likely that they convert than if they are from the US, holding all other variables constant.

3. If a user is from CA, it is 1.02 more likely that they convert than if they are from the US, holding all other variables constant.

P-values indicate that these variables do not have an impact on conversion.

As a final conclusion, we can not neglect the null hypothesis, and there is no need for the company to implement the new page.

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

```
Out[39]: 0
```