Analyze_ab_test_results_notebook

April 30, 2019

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

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

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

c. The number of unique users in the dataset.

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

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

d. The proportion of users converted.

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

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

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

```
In [6]: df[(df['landing_page'] == "old_page") & (df['group'] == "treatment")].count()[0] +  df[(
```

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

f. Do any of the rows have missing values?

```
In [7]: df.isnull().count()  # no rows miss values
```

```
Out[7]: user_id         294478
        timestamp       294478
        group           294478
        landing_page    294478
        converted       294478
        dtype: int64
```

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

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

```
In [8]: df2 = df[((df.group == 'control') & (df.landing_page=='old_page')) | ((df.group == 'trea
        df2.shape

Out[8]: (290585, 5)

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()]

Out[11]:       user_id                   timestamp      group landing_page  converted
         2893   773192  2017-01-14 02:55:59.590927  treatment     new_page          0

In [12]: # User id is 773192
```

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

```
In [13]: # row ID is 2893,  landing page is new page, group is treatment, converted value is 0

In [14]: df2[df2['user_id'] == 773192] # two rows with duplicated ID

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.sort_values("user_id", inplace = True)
         df2.drop_duplicates(subset="user_id", 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.
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:2: 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#
```

```
In [16]: df2[df2['user_id']==773192] # only one row left.
```

```
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['user_id'].nunique()
```

```
Out[17]: 290584
```

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

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

```
In [18]: df2.head()
```

```
Out[18]:         user_id                   timestamp      group landing_page  converted
         63114    630000  2017-01-19 06:26:06.548941  treatment     new_page          0
         103873   630001  2017-01-16 03:16:42.560309  treatment     new_page          1
         205236   630002  2017-01-19 19:20:56.438330    control     old_page          0
         247344   630003  2017-01-12 10:09:31.510471  treatment     new_page          0
         242283   630004  2017-01-18 20:23:58.824994  treatment     new_page          0
```

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

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

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

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

```
Out[20]: 0.1203863045004612
```

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

```
In [21]: df2[df2['group'] == "treatment"]['converted'].mean()
```

```
Out[21]: 0.11880806551510564
```

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

```
In [22]: df2[df2['landing_page'] == "new_page"].count()[0] /  df2.shape[0]
```

```
Out[22]: 0.50006194422266881
```

e. Use the results in the previous two portions of this question to suggest if you think there is evidence that one page leads to more conversions? Write your response below.

Answer: The users receiving old pages and new pages are half and half. The conversion rate of receiving old page is 12.0%. The conversion rate of receiving new page is 11.9% There is no evidence that one 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.

$$H_0 : p_{new} \leq p_{old}$$

$$H_1 : p_{new} > p_{old}$$

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 **convert rate** for p_{new} under the null?

```
In [23]: # assume p_new = p_old, assume they are equal to the converted rate in ab_data.csv rega

         p_new = df2['converted'].mean()
         print(p_new)
```

```
0.119597087245
```

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

```
In [24]: # p_old equal to p_new
         p_old = p_new
         print(p_old)
```

```
0.119597087245
```

c. What is n_{new}?

```
In [25]: # use the sample size equal to the ones in ab_data.csv
         n_new = len(df2[df2['landing_page'] == "new_page"])
         print(n_new)
```

```
145310
```

d. What is n_{old}?

```
In [26]: n_old = len(df2[df2['landing_page'] == "old_page"])
         print(n_old)
```

```
145274
```

e. Simulate n_{new} transactions with a convert rate of p_{new} under the null. Store these n_{new} 1's and 0's in **new_page_converted**.

```
In [27]: new_page_converted = np.random.choice(2, n_new, p=[1-p_new, p_new])
```

f. Simulate n_{old} transactions with a convert rate of p_{old} under the null. Store these n_{old} 1's and 0's in **old_page_converted**.

```
In [28]: old_page_converted = np.random.choice(2, n_old, p=[1-p_old, p_old])
```

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

```
In [29]: np.mean(new_page_converted) - np.mean(old_page_converted)
```

```
Out[29]: -2.0386773667280256e-06
```

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

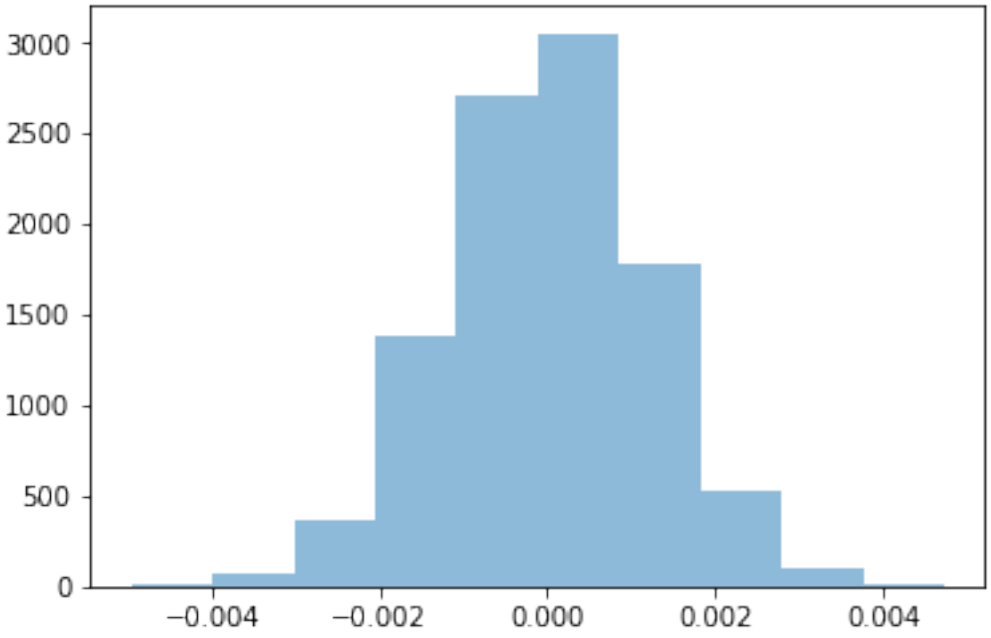
```
In [30]: p_diffs = []
         for _ in range(10000):
             new_temp = np.random.choice(2, n_new, p=[1-p_new, p_new]) # new page array
             old_temp = np.random.choice(2, n_old, p=[1-p_old, p_old]) # old page array
             p_diffs.append(np.mean(new_temp) - np.mean(old_temp))      # store the diffs in p_d

         np.mean(p_diffs)
```

```
Out[30]: 7.1909029414329791e-06
```

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 [31]: plt.hist(p_diffs, alpha = 0.5); # The plot is normal distribution as expected because c
```



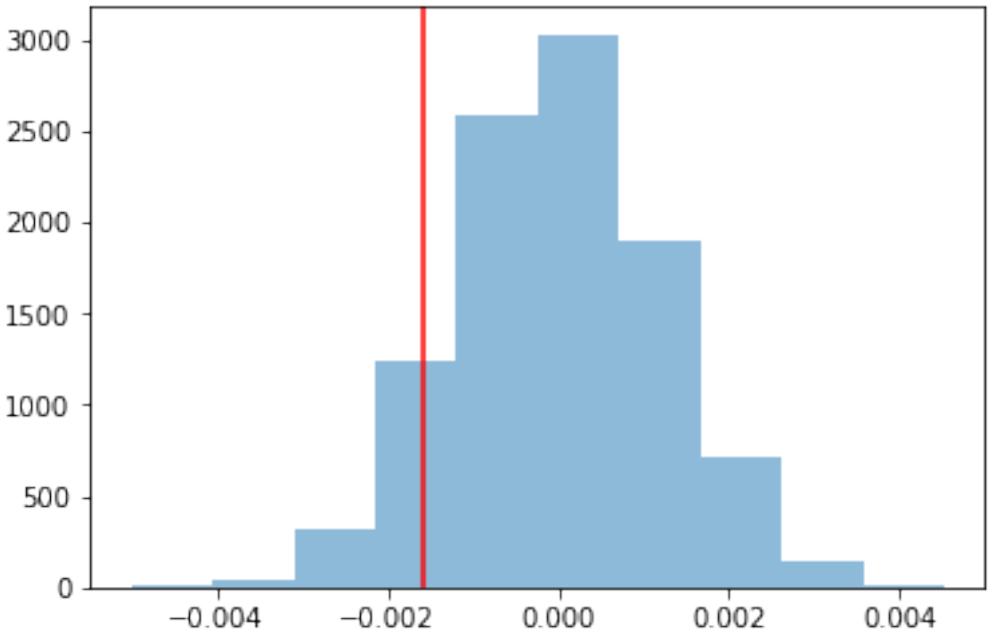
j. What proportion of the **p_diffs** are greater than the actual difference observed in **ab_data.csv**?

```
In [38]: # actual difference in ab_data.csv
         actual_diff = df2[df2['landing_page'] == "new_page"]['converted'].mean() - df2[df2['lan
         print(actual_diff)

         # plot the null_value
         null_vals = np.random.normal(0, np.std(p_diffs), 10000)

         plt.hist(null_vals, alpha=0.5);
         plt.axvline(x=actual_diff, color = 'red'); #  the areas to the right of the red bar are
```

```
-0.00157823898536
```




```
In [39]: # calculate the p_value

         p_value = (null_vals > actual_diff).mean()
         print(p_value)
```

```
0.9063
```

k. In words, explain what you just computed in part **j.**. What is this value called in scientific studies? What does this value mean in terms of whether or not there is a difference between the new and old pages?

Answer: this value is called p-value. If type I error rate threshold is 5%, this p-value, 0.91, is greater than type I error rate. It means we fail to reject the null hypothesis. Thus, the conversion rate of the old page is as good as it for the new 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 [40]: import statsmodels.api as sm

         convert_old = df2[df2['landing_page'] =="old_page"]['converted'].mean()
```

```
convert_new = df2[df2['landing_page'] =="new_page"]['converted'].mean()
n_old = len(df2[df2['landing_page'] == "old_page"])
n_new = len(df2[df2['landing_page'] == "new_page"])
```

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

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 [41]: import statsmodels.api as sm
         z_score, p_value = sm.stats.proportions_ztest([convert_old, convert_new], [n_old, n_new
         print(z_score)
         print(p_value)
```

```
0.00328757967535
0.99737689566
```

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.**?

Answer: I picked a significance level 95%. This is a one-tail test so a z-score past 1.95 will be significant. In our case, Z-score is 0.003, less than 1.95 critical value. Thus I fail to reject the null hypothesis. The old page is as good as the new page.

Part III - A regression approach

1. In this final part, you will see that the result you acheived in the previous A/B test can also be acheived by performing regression.

a. Since each row is either a conversion or no conversion, what type of regression should you be performing in this case?

Answer: This is Logistic Regression.

b. The goal is to use **statsmodels** to fit the regression model you specified in part **a.** to see if there is a significant difference in conversion based on which page a customer receives. However, you first need to create a column for the intercept, and create a dummy variable column for which page each user received. Add an **intercept** column, as well as an **ab_page** column, which is 1 when an individual receives the **treatment** and 0 if **control**.

```
In [42]: # create dummy variable
         df2[['treatment','ab_page']] = pd.get_dummies(df2['group'])
         df2 = df2.drop('treatment', axis = 1)
```

```
/opt/conda/lib/python3.6/site-packages/pandas/core/frame.py:2352: 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]
```

```
In [43]: df2.head()
```

```
Out[43]:         user_id                   timestamp      group landing_page  \
         63114    630000  2017-01-19 06:26:06.548941  treatment     new_page
         103873   630001  2017-01-16 03:16:42.560309  treatment     new_page
         205236   630002  2017-01-19 19:20:56.438330    control     old_page
         247344   630003  2017-01-12 10:09:31.510471  treatment     new_page
         242283   630004  2017-01-18 20:23:58.824994  treatment     new_page

                 converted  ab_page
         63114           0        1
         103873          1        1
         205236          0        0
         247344          0        1
         242283          0        1
```

```
In [44]: # add an intercept
         df2['intercept'] = 1
```

c. Use **statsmodels** to import your regression model. Instantiate the model, and fit the model using the two columns you created in part **b.** to predict whether or not an individual converts.

```
In [45]: df2['intercept']=1
         lm = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])
         results = lm.fit()
         results.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.366118
         Iterations 6
```

```
Out[45]: <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:                Wed, 16 Jan 2019   Pseudo R-squ.:               8.077e-06
         Time:                        02:06:30   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
```

```
==========================================================================
"""
```

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

Answer: conversion rate will decrease if changing the type of page, old page to new page or vice versa. The p-value of page type change is 0.19, bigger than 0.05. This means page type change does not have significate impact on conversion rate.

e. What is the p-value associated with **ab_page**? Why does it differ from the value you found in the **Part II**? **Hint**: What are the null and alternative hypotheses associated with your regression model, and how do they compare to the null and alternative hypotheses in the **Part II**?

Answer: ab_page's p-value is 1-0.190/2=0.9. The hypothesis of this regression model and Part II are different. In the regresion model, the concern is about whether the type of page in general impacts the conversion rate, thus a two-tailed test. It does not care of which type of page has more impact than the other. In part II, we are concerned about whether the rate of new page is more than old page, thus a one-tailed test. So they are different.

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?

Answer: Because the type of page does not have statistical evidence to impact the conversion rate, we need to consider other terms. I need to be careful that the added terms are independant to existing ones. For example, adding "country" is a good idea. But if the type of page is chosen based on country, introducing country is not a good idea.

g. Now along with testing if the conversion rate changes for different pages, also add an effect based on which country a user lives. You will need to read in the **countries.csv** dataset and merge together your datasets on the approporiate rows. Here are the docs for joining tables.

Does it appear that country had an impact on conversion? Don't forget to create dummy variables for these country columns - **Hint: You will need two columns for the three dummy varaibles.** Provide the statistical output as well as a written response to answer this question.

```
In [46]: # read country data
         df_c = pd.read_csv('countries.csv')
         df_c.head()
```

```
Out[46]:    user_id country
         0   834778      UK
         1   928468      US
         2   822059      UK
         3   711597      UK
         4   710616      UK
```

```
In [47]: # join tables
         df_join = df2.set_index('user_id').join(df_c.set_index('user_id'))
         df_join.head()
```

Out[47]:

user_id	timestamp	group	landing_page	converted	ab_page	intercept	country
630000	2017-01-19 06:26:06.548941	treatment	new_page	0	1	1	US
630001	2017-01-16 03:16:42.560309	treatment	new_page	1	1	1	US
630002	2017-01-19 19:20:56.438330	control	old_page	0	0	1	US
630003	2017-01-12 10:09:31.510471	treatment	new_page	0	1	1	US
630004	2017-01-18 20:23:58.824994	treatment	new_page	0	1	1	US

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

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

```
In [49]: # add dummy
         df_join[['CA','UK','US']] = pd.get_dummies(df_join['country'])
         df_join.head()
```

Out[49]:

user_id	timestamp	group	landing_page	converted	ab_page	intercept	country	CA	UK	US
630000	2017-01-19 06:26:06.548941	treatment	new_page	0	1	1	US	0	0	1
630001	2017-01-16 03:16:42.560309	treatment	new_page	1	1	1	US	0	0	1
630002	2017-01-19 19:20:56.438330	control	old_page	0	0	1	US	0	0	1
630003	2017-01-12 10:09:31.510471	treatment	new_page	0	1	1	US	0	0	1
630004	2017-01-18 20:23:58.824994	treatment	new_page	0	1	1	US	0	0	1

```
In [51]: # drop one dummy
         df_join = df_join.drop('CA', axis= 1)
         df_join.head()
```

Out[51]:

user_id	timestamp	group	landing_page	converted

```
630000   2017-01-19 06:26:06.548941  treatment     new_page          0
630001   2017-01-16 03:16:42.560309  treatment     new_page          1
630002   2017-01-19 19:20:56.438330    control     old_page          0
630003   2017-01-12 10:09:31.510471  treatment     new_page          0
630004   2017-01-18 20:23:58.824994  treatment     new_page          0

         ab_page  intercept country  UK  US
user_id
630000         1          1      US   0   1
630001         1          1      US   0   1
630002         0          1      US   0   1
630003         1          1      US   0   1
630004         1          1      US   0   1
```

```
In [52]: lm2 = sm.Logit(df_join['converted'], df_join[['intercept', 'ab_page', 'UK', 'US']])
         results2 = lm2.fit()
         results2.summary()
```

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

```
Out[52]: <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:                Wed, 16 Jan 2019   Pseudo R-squ.:               2.323e-05
         Time:                        02:12:47   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
         UK             0.0506      0.028      1.784      0.074      -0.005       0.106
         US             0.0408      0.027      1.516      0.130      -0.012       0.093
         ==============================================================================
         """
```

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.

Answer: The p-values for ab_page, UK and US indicate none of them has significant effects on conversion rate.

Finishing Up

Congratulations! You have reached the end of the A/B Test Results project! This is the final project in Term 1. You should be very proud of all you have accomplished!

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

```
Out[1]: 0
```

```
In [ ]:
```