

Analyze_ab_test_results_notebook

December 12, 2019

1 Mohamed Makki

1.1 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!

1.2 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 [29]: 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 [30]: df = pd.read_csv('ab_data.csv')
df.head()
```

```
Out[30]:
```

	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 [31]: df.shape[0]
```

```
Out[31]: 294478
```

c. The number of unique users in the dataset.

```
In [32]: df.user_id.nunique()
```

```
Out[32]: 290584
```

d. The proportion of users converted.

```
In [33]: df[df['converted'] == 1].shape[0] / df.user_id.shape[0]
```

```
Out[33]: 0.11965919355605512
```

```
In [34]: # or simply can be calculated by mean of converted users
df['converted'].mean()
```

```
Out[34]: 0.11965919355605512
```

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

```
In [35]: # we have two cases when new_page do not match treatement, we'll calculate each and the
```

```
case1 = df.query('landing_page == "new_page" and group == "control"').shape[0]
case2 = df.query('landing_page == "old_page" and group == "treatment"').shape[0]
no_match = case1 + case2
no_match
```

```
Out[35]: 3893
```

f. Do any of the rows have missing values?

```
In [36]: df.isnull().sum()
# as below there is no row with missing value
```

```
Out[36]: 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 [37]: # to create the new dataset df2 we should drop the 3893 rows calculated above.
```

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

```
In [38]: # now we'll rename our new data [df2] (after dropping the 3893 rows that the new_page a
df2 = df
```

```
In [39]: # Double Check all of the correct rows were removed - this should be 0
df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].shape
```

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

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

a. How many unique **user_ids** are in **df2**?

```
In [40]: df2['user_id'].nunique()
```

```
Out[40]: 290584
```

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

```
In [41]: df2[df2['user_id'].duplicated()]
# we have only one duplicated user_id in df2 as below
```

```
Out[41]:
```

	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 [42]: df2[df2['user_id']== 773192 ]
```

```
Out[42]:
```

	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 [43]: df2.drop_duplicates('user_id', inplace=True)
```

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

```
Out[44]: 0.11959708724499628
```

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

```
In [45]: df2.query('group == "control"').converted.mean()
```

```
Out[45]: 0.1203863045004612
```

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

```
In [46]: df2.query('group == "treatment"').converted.mean()
```

```
Out[46]: 0.11880806551510564
```

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

```
In [47]: (df2['landing_page'] == 'new_page').mean()
```

```
Out[47]: 0.50006194422266881
```

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.

Your answer goes here. ### from the above data, the conversion rate from control to treatment group (0.1204 = %12.04) ### in the other hand the conversion rate from treatment to control group (0.1188 = %11.88) ### then both of them have the same rate of conversion about 12% (or the conversion rate to treatment group > the conversion rate to control group by about (%0.16) which so small. ### we should advise to keep the current pages as the new versions would not make any significant change. ### in other words we have NO tangible evidence to reject the current version.

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.

H0 : $p_{new} - p_{old} \leq 0$ fail to reject the null hypothesis

H1 : $p_{new} - p_{old} > 0$ reject the null hypothesis and choose the alternative 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 [48]: p_new = df2.converted.mean()  
p_new
```

```
Out[48]: 0.11959708724499628
```

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

```
In [49]: p_old = df2.converted.mean()  
p_old
```

```
Out[49]: 0.11959708724499628
```

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

```
In [50]: n_new = df2.query('group == "treatment").count()[0]  
n_new
```

```
Out[50]: 145310
```

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

```
In [51]: n_old = df2.query('group == "control").count()[0]  
n_old
```

```
Out[51]: 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 [59]: new_page_converted = np.random.choice([0, 1], n_new, p = [p_new, 1 - p_new])  
new_page_converted
```

```
Out[59]: array([1, 1, 1, ..., 0, 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 [60]: old_page_converted = np.random.choice([0, 1], n_old, p = [p_old, 1 - p_old])
         old_page_converted
```

```
Out[60]: array([1, 1, 1, ..., 1, 0, 1])
```

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

```
In [61]: p_diff = new_page_converted.mean() - old_page_converted.mean()

         p_diff
```

```
Out[61]: 0.0012545418708010958
```

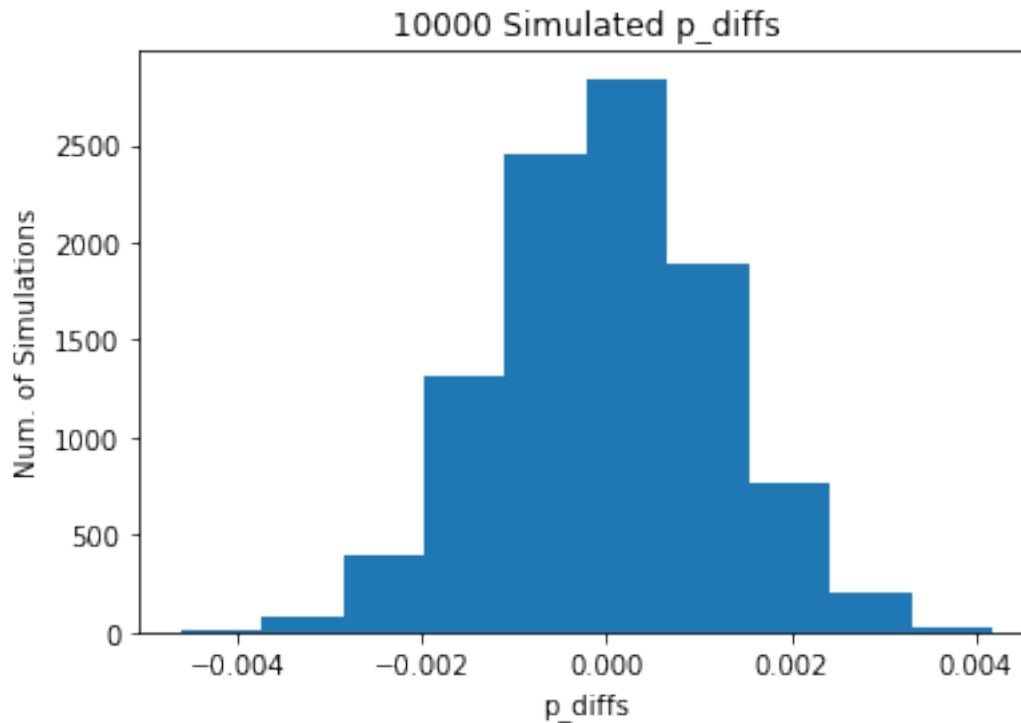
- h. Create 10,000 $p_{new} - p_{old}$ values using the same simulation process you used in parts (a) through (g) above. Store all 10,000 values in a NumPy array called **p_diffs**.

```
In [68]: p_diffs = []

         for _ in range(10000):
             new_page_converted = np.random.choice([0, 1], n_new, p = [p_new, 1 - p_new]).mean()
             old_page_converted = np.random.choice([0, 1], n_old, p = [p_old, 1 - p_old]).mean()
             p_diffs.append(new_page_converted - old_page_converted)
```

- 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 [69]: plt.hist(p_diffs);
         plt.ylabel('Num. of Simulations');
         plt.xlabel('p_diffs');
         plt.title('10000 Simulated p_diffs');
```



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

In [70]: *#computing the actual difference of converted rates*

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

actual_diff = actual_mean_new - actual_mean_old
actual_diff
```

Out[70]: -0.0015782389853555567

In [71]: *# p_diff array*

```
p_diffs = np.array(p_diffs)
```

In [72]: *#computing proportion of the p_diffs are greater than the actual difference observed in*

```
(p_diffs > actual_diff).mean()
```

Out[72]: 0.905800000000000005

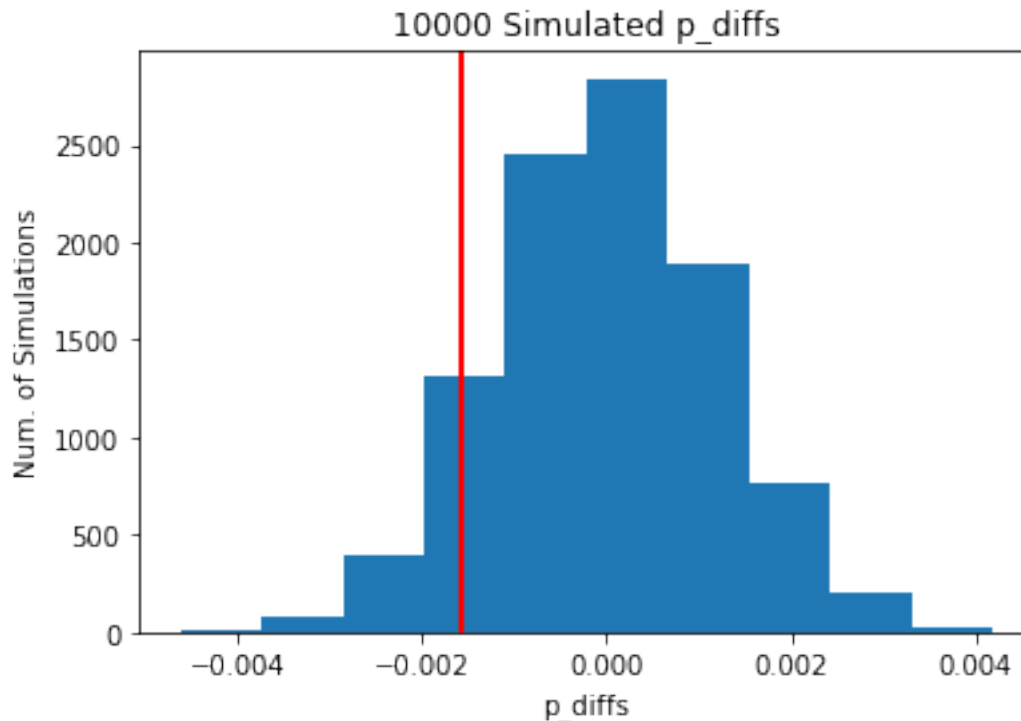
In [73]: `plt.hist(p_diffs);`

```
plt.ylabel('Num. of Simulations');
```

```
plt.xlabel('p_diffs');
```

```
plt.title('10000 Simulated p_diffs');
```

```
plt.axvline(actual_diff,color='r',linewidth = 2);
```



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?

1.2.1 we have alpha-value () that represents the type I error = 0.05

1.2.2 actual_diff: represents the difference calculated based on our dataset

1.2.3 p_diff : represents simulated difference based on 10000 simulated samples

1.2.4 the final difference = p_value

1.2.5 p_value = 0.91

1.2.6 p_value > : because $0.91 > 0.05$. . (the p_value is very high and much bigger than the allowed alpha value)

1.2.7 accordingly there is no significant evidence to reject the null hypothesis

1.2.8 fail to reject the null hypothesis . . keep the old pages unchanged.

1. 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 [75]: 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('group == "control"')['user_id'].count()
n_new = df2.query('group == "treatment"')['user_id'].count()

convert_old, convert_new, n_old, n_new
```

```
Out[75]: (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 [88]: z_score, p_value = sm.stats.proportions_ztest([convert_old, convert_new], [n_old, n_new])
        z_score, p_value
```

```
Out[88]: (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.?

1.2.9 p_value = 0.905 same p_value as calculated above.

1.2.10 z-score represents standard deviation of mean population, negative z-score: reveals that the raw score is below the mean average so (z-score = +1.3109 indicates it is 1.3109 standard deviation above the mean)

1.2.11 z-score is between -1.96 and +1.96 $-1.96 < (z\text{-score}) < +1.96$

1.2.12 accordingly the confidence level = %95

1.2.13 this confidence level reveals that p-value will be larger than 0.05 (allowed type I error)

1.2.14 accordingly we can not reject the null hypothesis

1.2.15 again fail to reject the null hypothesis . . . the same conclusion we reached before in part 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?

1.2.16 the proper choice here 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 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 [89]: df2['intercept']=1
df2[['ab_page', 'treatment']]=pd.get_dummies(df2['group'])
df2.drop('treatment', axis=1, inplace=True)
df2.head()
```

```
Out[89]:
```

	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	1
1	1	1
2	1	0
3	1	0
4	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 [90]: logit_mod = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])
```

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

```
In [91]: results=logit_mod.fit()
results.summary()
```

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

```
Out[91]: <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:                            Thu, 12 Dec 2019    Pseudo R-squ.:                   8.077e-06
Time:                            12:52:43      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	-2.0038	0.008	-247.146	0.000	-2.020	-1.988
ab_page	0.0150	0.011	1.311	0.190	-0.007	0.037
=====						

- 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 = 0.190, it differs from the value calculated in Part II above (= 0.91) because of the effect of the intercept and adding new categorical columns values to our dataset.

- 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?

Adding or introducing more variables to our model may make it better when introducing the right variables and this may increase the accuracy and make the model a better model. but this will depend on whether the current model has the right numbers of variables or not, in other hand there are some disadvantages in adding more variables to the regression model, it may make our model more complex and then increase the chance of an over-fit.

- 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 [92]: countries_df = pd.read_csv('./countries.csv')
df3 = countries_df.set_index('user_id').join(df2.set_index('user_id'), how='inner')
df3.head()
```

```
Out[92]:
```

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


```

converted  intercept  ab_page
user_id
```

834778	0	1	1
928468	0	1	0
822059	1	1	0
711597	0	1	1
710616	0	1	0

```
In [93]: # defining the countries included in our dataset
df3.country.unique()
```

```
Out[93]: array(['UK', 'US', 'CA'], dtype=object)
```

```
In [94]: old_converted_UK = df3.query('group == "control" and country == "UK"').converted.mean()
old_converted_US = df3.query('group == "control" and country == "US"').converted.mean()
old_converted_CA = df3.query('group == "control" and country == "CA"').converted.mean()

old_converted_UK, old_converted_US, old_converted_CA
```

```
## In the old pages, the figures below shows:
## There are no tangible differences in conversion in different countries,
## it is clear that the country has no impact on the conversion (conversion rate about
```

```
Out[94]: (0.12002200220022002, 0.12062998938220143, 0.11878299527646569)
```

```
In [95]: new_converted_UK = df3.query('group == "treatment" and country == "UK"').converted.mean()
new_converted_US = df3.query('group == "treatment" and country == "US"').converted.mean()
new_converted_CA = df3.query('group == "treatment" and country == "CA"').converted.mean()

new_converted_UK, new_converted_US, new_converted_CA
```

```
## In the new pages, the figures below shows:
## There are no tangible differences in conversion in different countries,
## it is clear that the country has no impact on the conversion (conversion rate about
```

```
Out[95]: (0.1211709965102753, 0.11846559963887227, 0.11190247911245035)
```

```
In [100]: # adding new columns for US & Uk
df3[['US', 'UK']] = pd.get_dummies(df3['country'])[['US', 'UK']]
df3.head()
```

```
Out[100]:
```

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

	converted	intercept	ab_page	US	UK
user_id					

834778	0	1	1	0	1
928468	0	1	0	1	0
822059	1	1	0	0	1
711597	0	1	1	0	1
710616	0	1	0	0	1

In [103]: *# Creating Fit model and summary*

```
log_mod = sm.Logit(df3['converted'], df3[['intercept', 'ab_page', 'US', 'UK']])
results = log_mod.fit()
results.summary()
```

Optimization terminated successfully.

Current function value: 0.366113

Iterations 6

Out[103]: <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:                        Thu, 12 Dec 2019    Pseudo R-squ.:                2.323e-05
Time:                        13:00:31          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                   -2.0450      0.027     -76.820      0.000     -2.097     -1.993
ab_page                      0.0149      0.011      1.307      0.191     -0.007      0.037
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
=====
"""
```

- 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 [106]: *# from the coefficient in the table.*

```
# ab_page coef = 0.0149
```

```
# US-users: coef = 0.0408
```

```
# UK-users: coef = 0.0506
```

```
np.exp(0.0149), np.exp(0.0408), np.exp(0.0506)
```

```
Out[106]: (1.0150115583846535, 1.0416437559600236, 1.0519020483004984)
```

from the above results: for the ab_page: if the individual is in the new page, the conversion rate to old page (1.015) more likely than he uses the old page, holding all else constant. for the US-user: conversion rate (1.04) times more likely than if he is a CA-user, holding all else constant for the UK_user: conversion rate (1.05) times more likely than if he is a CA-user, holding all else constant the p_ values for both countries (0.130 & 0.074) respectively, both of them are greater than alpha value (0.05) accordingly they are both statistically significant, this will show us that the conversion rate in each country is same as in all users, the conclusion is that all values are statistically significant and we fail to reject the null hypothesis. we will keep our current pages.

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 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.

1.3 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** sub-menu, 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 [107]: from subprocess import call
          call(['python', '-m', 'nbconvert', 'Analyze_ab_test_results_notebook.ipynb'])
```

```
Out[107]: 255
```