Analyze_ab_test_results_notebook

July 1, 2021

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
- Part I - Probability
- Part II - A/B Test
- Part III - Regression

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.

[45]:
```
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`.

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

[46]:
```
df = pd.read_csv('ab_data.csv')
df.head()
```

[46]:

	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.

[47]:
```
default_df = df.shape[0]
default_df
```

[47]:
```
294478
```

   c. The number of unique users in the dataset.

[48]:
```
df['user_id'].nunique()
```

[48]:
```
290584
```

   d. The proportion of users converted.

[49]:
```
df['converted'].mean()
```

[49]:
```
0.11965919355605512
```

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

[50]:
```
treat_old = df.query('group == "treatment" & landing_page != "new_page"')
control_new = df.query('group != "treatment" & landing_page == "new_page"')
len(treat_old) + len(control_new)
```

[50]:
```
3893
```

   f. Do any of the rows have missing values?

[51]:
```
print(df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 294478 entries, 0 to 294477
Data columns (total 5 columns):
```

```
 #   Column        Non-Null Count   Dtype
---  ------        --------------   -----
 0   user_id       294478 non-null  int64
 1   timestamp     294478 non-null  object
 2   group         294478 non-null  object
 3   landing_page  294478 non-null  object
 4   converted     294478 non-null  int64
dtypes: int64(2), object(3)
memory usage: 11.2+ MB
None
```

No missing values in this df

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.

   a. We drop these rows. We should only use the rows that we can feel confident in the accuracy of the data.

[52]:
```
df.drop(treat_old.index, axis=0, inplace=True)
df.drop(control_new.index, axis=0, inplace=True)
df2 = df
```

[53]:
```
# Checking
df2.shape[0] + 3893 == default_df
```

[53]:
```
True
```

[54]:
```
# Double Check all of the correct rows were removed - this should be 0
df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) ==␣
 ↪False].shape[0]
```

[54]:
```
0
```

3. Use **df2** and the cells below to answer questions.

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

[55]:
```
df2['user_id'].nunique()
```

[55]:
```
290584
```

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

[56]:
```
df2[df2.duplicated('user_id')].user_id
```

[56]:
```
2893    773192
Name: user_id, dtype: int64
```

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

[57]:
```
df2[df2['user_id'] == 773192]
```

[57]:

	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**.

[58]:
```
df2.drop_duplicates('user_id', inplace=True)
```

[59]:
```
df2[df2['user_id'] == 773192]
```

[59]:

	user_id	timestamp	group	landing_page	converted
1899	773192	2017-01-09 05:37:58.781806	treatment	new_page	0

4. Use **df2** in the cells below to answer questions.

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

[60]:
```
df2['converted'].mean()
```

[60]: 0.11959708724499628

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

[61]:
```
control_convert = df2.query("group == 'control'")['converted'].mean()
control_convert
```

[61]: 0.1203863045004612

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

[62]:
```
treatment_convert = df2.query("group == 'treatment'")['converted'].mean()
treatment_convert
```

[62]: 0.11880806551510564

[63]:
```
obs_diffs = treatment_convert - control_convert
obs_diffs
```

[63]: -0.0015782389853555567

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

[64]:
```
df2.query("landing_page == 'new_page'").group.count()/df2['landing_page'].
↪count()
```

[64]: 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.

Taking into account the results above, we do not have enough information which page leads to more conversions. In addition we got very close results. To get more reliable

results, we would simulate results or create sampling distribution to get a probability distribution of a statistic obtained from a larger number of samples drawn from a specific population.

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.

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

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

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.

a. What is the **conversion rate** for p_{new} under the null?

[65]:
```
p_new0 = df2["converted"].mean()
p_new0
```

[65]:
```
0.11959708724499628
```

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

[66]:
```
p_old0 = p_new0
p_old0
```

[66]:
```
0.11959708724499628
```

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

[67]:
```
n_new = df2.query("group == 'treatment'").count()['group']
n_new
```

[67]:
```
145310
```

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

[68]:
```
n_old = df2.query("group == 'control'").count()['group']
n_old
```

[68]:
```
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**.

[69]:
```
new_page_converted = np.random.choice([0, 1], size = n_new, p = [1 - p_new0,␣
↪p_new0]).mean()
new_page_converted
```

[69]:
```
0.12048723418897529
```

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**.

[70]:
```
old_page_converted = np.random.choice([0, 1], size = n_old, p = [1 - p_old0,␣
↪p_old0]).mean()
old_page_converted
```

[70]:
```
0.11956716274075196
```

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

[71]:
```
new_page_converted - old_page_converted
```

[71]:
```
0.000920071448223328
```

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**.

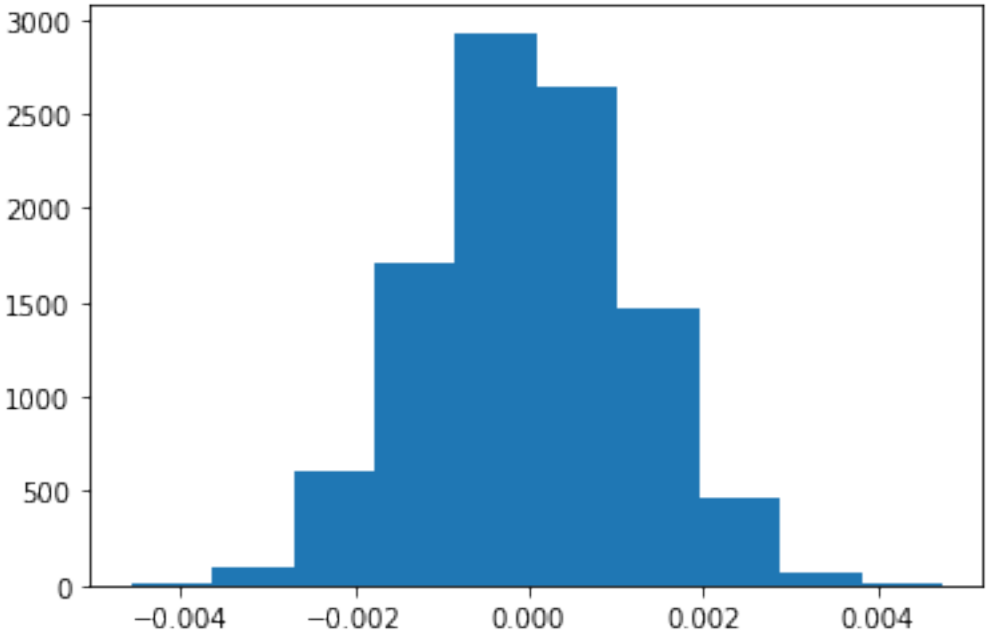
[72]:
```
p_diffs = []

for i in range(10000):
    new_page_converted = np.random.choice([0, 1], size = n_new, p = [1 -␣
↪p_new0, p_new0]).mean()
    old_page_converted = np.random.choice([0, 1], size = n_old, p = [1 -␣
↪p_old0, p_old0]).mean()
    p_diffs.append(new_page_converted - old_page_converted)
```

[73]:
```
p_diffs = np.array(p_diffs)
```

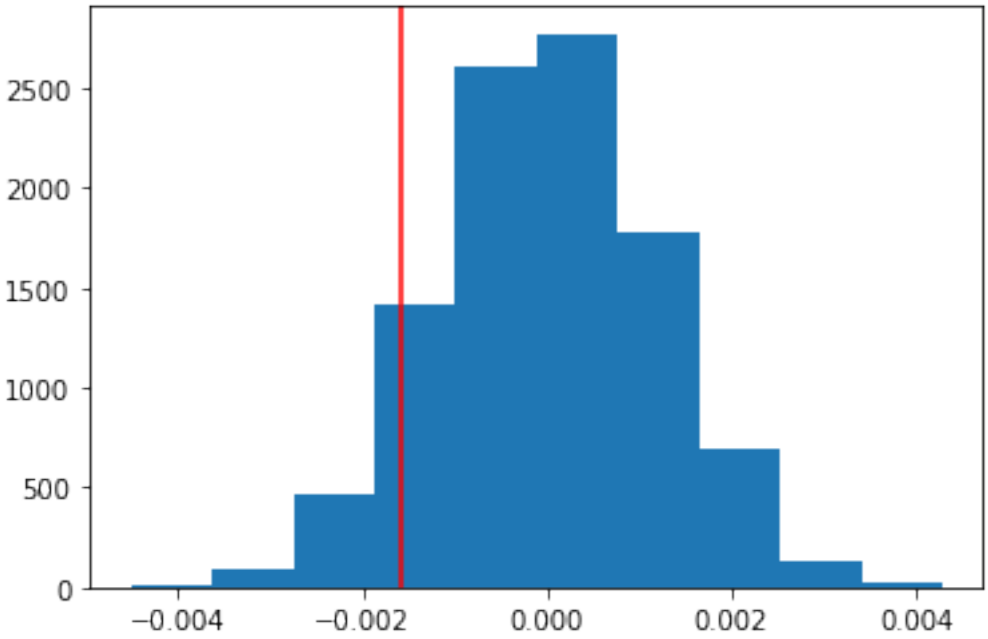
i. Plot a histogram of the **p_diffs**. Does this plot look like what you expected? Use the matching problem in the classroom to assure you fully understand what was computed here.

[74]:
```
plt.hist(p_diffs);
```



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

[75]:
```
null_vals = np.random.normal(0, np.std(p_diffs), 10000)
plt.hist(null_vals);
plt.axvline(x=obs_diffs, color = 'red');
```



[76]:
```
p_val = (null_vals > obs_diffs).mean()
p_val
```

[76]: 0.9073

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?

In part j i calculated p-value. The p-value is the probability of obtaining our observed statistic or a "more extreme" value if the null hypothesis is true. The p-value could be any value appropriate to the situation. At low p-values(usually 0.05) we typically reject the null hypothesis. In our case, p-value more than 0.05 then we fail to reject H_0

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.

[77]:
```
import statsmodels.api as sm

convert_old = df2.query("landing_page == 'old_page'").converted.sum()
convert_new = df2.query("landing_page == 'new_page'").converted.sum()
n_old = df2.query("landing_page == 'old_page'").user_id.count()
n_new = df2.query("landing_page == 'new_page'").user_id.count()
```

```
print(convert_old, convert_new, n_old, n_new,)
```

```
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.

[78]:
```
from statsmodels.stats.proportion import proportions_ztest
```

[79]:
```
num_of_succes = np.array([convert_old, convert_new])
tot_sample_size = np.array([n_old, n_new])

z_score, p_value = sm.stats.proportions_ztest(num_of_succes, tot_sample_size,␣
↪alternative='smaller')
print('z-score' , z_score)
print('p-value' , p_value)
```

```
z-score 1.3109241984234394
p-value 0.9050583127590245
```

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.**?

z-score it is a measure of the relative spread of the observed value, which indicates how many deviations constitute its spread in the relative mean. Here i got similar p-value(0.905) like in part j, so i said we fail to reject H_0

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?

In this situation we will use the logical 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**.

[80]:
```
# Creating an intercept column
df['intercept'] = 1

# Creating dummy variables
df2[['ab_page']] = pd.get_dummies(df2['group'])['treatment']
df2.head()
```

[80]:
```
   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.

[81]:
```
log_mod = sm.Logit(df['converted'], df[['intercept', 'ab_page']])
# Fit() our results
results = log_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.

[82]:
```
results.summary()
```

[82]:
```
<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, 01 Jul 2021   Pseudo R-squ.:               8.077e-06
Time:                        11:28:21   Log-Likelihood:            -1.0639e+05
converged:                       True   LL-Null:                   -1.0639e+05
Covariance Type:            nonrobust   LLR p-value:                    0.1899
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -1.9888      0.008   -246.669      0.000      -2.005      -1.973
ab_page       -0.0150      0.011     -1.311      0.190      -0.037       0.007
==============================================================================
"""
```

e. What is the p-value associated with **ab_page**? Why does it differ from the value you found in **Part II**?

The p-value is 0.190. It means we fail to reject H_0 and this variable is not statistically significant. In logical regression model here we only consider is either a conversion or no conversion. But in Part II we considered less than or equal and greater than.

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?

The more factors or data in regression model, the more accureate our results. Because of this undoubtedly good idea to add other factors. But if we add big quantity of data it makes our analysis more complicated and hard readable. And this in turn can lead our data to multicollinearity, it is when our variables are correlated with one another.

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. Provide the statistical output as well as a written response to answer this question.

[83]:
```
countries_df = pd.read_csv('countries.csv')
countries_df.head()
```

[83]:

	user_id	country
0	834778	UK
1	928468	US
2	822059	UK
3	711597	UK
4	710616	UK

[84]:
```
# Joinung two dataframes
new_model =  df2.join(countries_df.set_index('user_id'), on = 'user_id')
new_model.head()
```

[84]:

	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
```

[85]:
```
# Creating dummy variables
new_model[['CA', 'UK', 'US']] = pd.get_dummies(new_model['country'])
new_model.head()
```

[85]:
```
   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  CA  UK  US
0          1        0      US   0   0   1
1          1        0      US   0   0   1
2          1        1      US   0   0   1
3          1        1      US   0   0   1
4          1        0      US   0   0   1
```

[86]:
```
log_mod2 = sm.Logit(new_model['converted'], new_model[['intercept', 'CA',␣
↪'US']])
# Fit() our results
results2 = log_mod2.fit()
results2.summary()
```

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

[86]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              converted   No. Observations:               290584
Model:                          Logit   Df Residuals:                   290581
Method:                           MLE   Df Model:                            2
Date:                Thu, 01 Jul 2021   Pseudo R-squ.:               1.521e-05
Time:                        11:28:25   Log-Likelihood:            -1.0639e+05
converged:                       True   LL-Null:                   -1.0639e+05
Covariance Type:            nonrobust   LLR p-value:                    0.1984
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -1.9868      0.011   -174.174      0.000      -2.009      -1.964
CA            -0.0507      0.028     -1.786      0.074      -0.106       0.005
US            -0.0099      0.013     -0.746      0.456      -0.036       0.016
```

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

The p-value for all countries in the model above is greater than 0.05. It is mean that this variables are not statistically significant for our model and we fail to reject H_0.

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.

[87]:
```
new_model['US_ind_ab_page'] = new_model['US'] * new_model['ab_page']
new_model['CA_ind_ab_page'] = new_model['CA'] * new_model['ab_page']

new_model.head()
```

[87]:

	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	CA	UK	US	US_ind_ab_page	CA_ind_ab_page
0	1	0	US	0	0	1	0	0
1	1	0	US	0	0	1	0	0
2	1	1	US	0	0	1	1	0
3	1	1	US	0	0	1	1	0
4	1	0	US	0	0	1	0	0

[88]:
```
log_mod2 = sm.Logit(new_model['converted'], new_model[['intercept', 'CA', 'US',␣
↪'ab_page', 'US_ind_ab_page', 'CA_ind_ab_page']])
# Fit() our results
results2 = log_mod2.fit()
results2.summary()
```

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

[88]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              converted   No. Observations:               290584
Model:                          Logit   Df Residuals:                   290578
Method:                           MLE   Df Model:                            5
Date:                Thu, 01 Jul 2021   Pseudo R-squ.:               3.482e-05
```

```
Time:                         11:28:29   Log-Likelihood:            -1.0639e+05
converged:                        True   LL-Null:                   -1.0639e+05
Covariance Type:             nonrobust   LLR p-value:                    0.1920
==============================================================================
==
                     coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
--
intercept         -1.9922      0.016   -123.457      0.000      -2.024
-1.961
CA                -0.0118      0.040     -0.296      0.767      -0.090
0.066
US                 0.0057      0.019      0.306      0.760      -0.031
0.043
ab_page            0.0108      0.023      0.475      0.635      -0.034
0.056
US_ind_ab_page    -0.0314      0.027     -1.181      0.238      -0.084
0.021
CA_ind_ab_page    -0.0783      0.057     -1.378      0.168      -0.190
0.033
==============================================================================
==
"""
```

An interaction between page and country did not effect to the any p-values. So that our conclusion remains the same

0.3 Conclusion

After completing our analysis and obtaining AB-test results we can say that we do not have evidence that our new page is brings more conversions than our old page. The variables we have are not statistically significant to influence the results. Because of this i would leave the old page or continiue improving the new one.

Finishing Up

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

0.4 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!

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