Analyze A/B Test Results

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!

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 (https://review.udacity.com/#!/projects/37e27304-ad47-4eb0-a1ab-8c12f60e43d0/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 a
        s 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('./data/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]: print("The number of rows in the dataset is:  " + str(df.shape[0]))

        The number of rows in the dataset is:  294478
```

c. The number of unique users in the dataset.

```
In [4]: print("The number of unique users in the dataset is:  " + str(df['user_i
        d'].nunique()))

        The number of unique users in the dataset is:  290584
```

d. The proportion of users converted.

```
In [5]: converted_users = df[df['converted']==1]
        #converted_users.head()

        df['converted'].count()

        proportion_converted = round(converted_users['user_id'].nunique()/df['us
        er_id'].nunique(),2)
        print('The proportion of converted users is: ' +  str(proportion_convert
        ed ))

        The proportion of converted users is: 0.12
```

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

In [6]:
```
# Check when group = treatment or landing_page = new_page but not both

treatment_but_not_new = df[(df['group'] == 'treatment') & (df['landing_p
age'] != "new_page")]
new_but_not_treatment = df[(df['group'] != 'treatment') & (df['landing_p
age'] == "new_page")]
dont_line_up_1 = pd.concat([treatment_but_not_new,new_but_not_treatment
])
print('The number of times the new_page and treatment don\'t line up i
s:' + str(dont_line_up_1.shape[0]))
dont_line_up_1.head()
```

```
The number of times the new_page and treatment don't line up is:3893
```

Out[6]:

	user_id	timestamp	group	landing_page	converted
308	857184	2017-01-20 07:34:59.832626	treatment	old_page	0
327	686623	2017-01-09 14:26:40.734775	treatment	old_page	0
357	856078	2017-01-12 12:29:30.354835	treatment	old_page	0
685	666385	2017-01-23 08:11:54.823806	treatment	old_page	0
713	748761	2017-01-10 15:47:44.445196	treatment	old_page	0

f. Do any of the rows have missing values?

In [7]:
```
print('Do any of the rows have missing values?  Answer:   ' + str(df.isnu
ll().values.any()))
```

```
Do any of the rows have missing values?  Answer:  False
```

`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]: aligned_results_1 = df[(df['group'] == 'treatment') & (df['landing_page'
        ] == "new_page")]
        aligned_results_2 = df[(df['group'] != 'treatment') & (df['landing_page'
        ] != "new_page")]
        df2 = pd.concat([aligned_results_1,aligned_results_2])
        df2.head()
```

Out[8]:

	user_id	timestamp	group	landing_page	converted
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
6	679687	2017-01-19 03:26:46.940749	treatment	new_page	1
8	817355	2017-01-04 17:58:08.979471	treatment	new_page	1
9	839785	2017-01-15 18:11:06.610965	treatment	new_page	1

```
In [9]: # Double Check all of the correct rows were removed - this should be 0
        df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page'
        )) == False].shape[0]

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]: print("The number of unique user_ids in the new dataset is:  " + str(len
         (df2['user_id'].unique())))
         print('Total number of user_ids:  ' + str(df2.shape[0]))

         The number of unique user_ids in the new dataset is:  290584
         Total number of user_ids:  290585
```

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

```
In [11]: df3 = df2[df2['user_id'].duplicated()]
         print('The one user_id that is repeated is:  ' + str(df3.iloc[0][0]))

         The one user_id that is repeated is:  773192
```

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

```
In [12]: print("The following is the row information for the repeat user_id:  ")
         df3.head()
```

```
The following is the row information for the repeat user_id:
```

Out[12]:

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

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

```
In [13]: df2.drop_duplicates(subset=['user_id'], keep='first',inplace=True)
         df2.shape

Out[13]: (290584, 5)
```

`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 [14]: total_prob_of_conv=df2['converted'].mean()
         print('The probability of an individual converting regardless of page i
         s:  ' + str(total_prob_of_conv))
```

```
The probability of an individual converting regardless of page is:  0.1
1959708724499628
```

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

```
In [15]: cg_u= df[df['group']=='control']
         cg_u_prob_of_conv = cg_u['converted'].mean()

         print('The probability of a control group user converting is:  ' + str(c
         g_u_prob_of_conv))
```

```
The probability of a control group user converting is:  0.1203991793589
7611
```

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

```
In [16]: tg_u= df[df['group']=='treatment']
         tg_u_prob_of_conv = tg_u['converted'].mean()

         print('The probability of a treatment group user converting is:  ' + str
         (tg_u_prob_of_conv))
```

```
The probability of a treatment group user converting is:  0.11891957956
489856
```

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

```
In [17]: r_np= df[df['landing_page']=='new_page']
         p_np=r_np['landing_page'].count()/df['landing_page'].count()
         print('The probability that an individual received a new page is:  ' + s
         tr(p_np))
```

```
The probability that an individual received a new page is:  0.5
```

e. Consider your results from a. through d. above, and explain below whether you think there is sufficient evidence to say that the new treatment page leads to more conversions.

Your answer goes here.

Answer: There is only a 0.0015 difference in conversions between the treatment page and the control page and the probability of the treatment group converting is slightly lower so it seems like there IS NOT sufficient evidence to say that the new treatment page leads to more conversions. Would need to do a statistical analysis to confirm.

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.

Put your answer here.

$$H_0 : p_{new} - p_{old} <= 0$$

$$H_1 : p_{new} - p_{old} > 0$$

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 [18]:
```
p_new=df2['converted'].mean()
print("p_new under null:  " + str(p_new))
```

```
p_new under null:  0.11959708724499628
```

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

In [19]:
```
p_old=p_new
print(p_old)
```

```
0.11959708724499628
```

c. What is n_{new}?

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

```
145310
```

d. What is n_{old}?

In [21]:
```
n_old=df2[df2['group']=='control'].shape[0]
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 [22]: new_page_converted = np.random.choice(2,n_new,p=[1-p_new,p_new])
         print(new_page_converted)

         [0 0 1 ..., 0 0 0]
```

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 [23]: old_page_converted = np.random.choice(2,n_old,p=[1-p_old,p_old])
         print(old_page_converted)

         [0 0 1 ..., 0 0 0]
```

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

```
In [24]: new_page_converted.mean()-old_page_converted.mean()

Out[24]: -0.0013512338048800587
```

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 a numpy array called **p_diffs**.

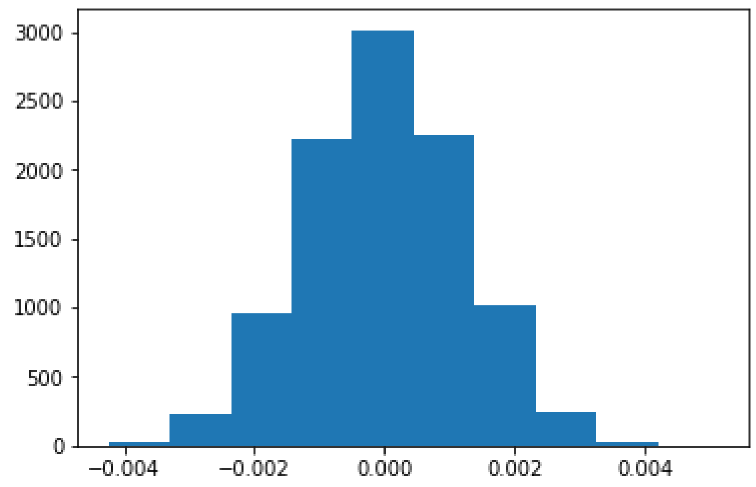
```
In [25]: print("This is the beginning")
         p_diffs = []
         for _ in range(10000):
             new_page_c = np.random.choice(2,n_new,p=[1-p_new,p_new])
             old_page_c = np.random.choice(2,n_old,p=[1-p_old,p_old])
             p_diffs.append(new_page_c.mean()-old_page_c.mean())
         print("This is the end")  # added print statements to see if it's finish
         ed

         This is the beginning
         This is the end
```

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.

Looks like a normal distribution

In [26]:
```
plt.hist(p_diffs);
```



In [27]:
```
print(np.percentile(p_diffs, 2.5),np.percentile(p_diffs,98.5))
print(np.percentile(p_diffs, 95))
print(np.percentile(p_diffs, 5))
print(np.percentile(p_diffs, 90)) # <-- results in 0.0015 which is about
 the same as calculated below
```

```
-0.00238332540618 0.00262716368788
0.00200065981407
-0.00199187225531
0.00156050078689
```

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

In [28]:
```
new_pages = df2[df2['group'] == 'treatment']
new_pages.head()
npc=new_pages['converted'].mean()
print("Percentage of new pages converted:  " + str(npc))
old_pages = df2[df2['group'] == 'control']
old_pages.head()
opc=old_pages['converted'].mean()
print("Percentage of old pages converted:  " + str(opc))
t_diff=npc-opc
print("Actual difference from data(p_new-p_old):  " + str(t_diff))
counter = 0
for i in range(len(p_diffs)):
    if p_diffs[i] > t_diff:
        counter = counter + 1
proportion_higher = counter/len(p_diffs)
print(proportion_higher)
```

```
Percentage of new pages converted:  0.11880806551510564
Percentage of old pages converted:  0.1203863045004612
Actual difference from data(p_new-p_old):  -0.0015782389853555567
0.9024
```

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?

Put your answer here.

~90% of p-diffs are greater than the actual observed difference in the data. This is the p-value.

We can **accept** the null because the difference is not outside the p-critical (0.05)

We would want 95% of the values to be above the observed difference to be sure that the number of possible Type-1 errors would be less than 5%.

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 [29]:
```
import statsmodels.api as sm

convert_old = df2.query("group == 'control' and converted == 1").count()
[0]
convert_new = df2.query("group == 'treatment' and converted == 1").count
()[0]
n_old = df2[df2['group'] == 'control'].count()[0]
n_new = df2[df2['group'] == 'treatment'].count()[0]
print(convert_old)
print(convert_new)
print(n_old)
print(n_new)
```

```
/Users/zoemclaughlin/anaconda3/lib/python3.6/site-packages/statsmodels/
compat/pandas.py:56: FutureWarning: The pandas.core.datetools module is
deprecated and will be removed in a future version. Please use the pand
as.tseries module instead.
  from pandas.core import datetools

17489
17264
145274
145310
```

m. Now use `stats.proportions_ztest` to compute your test statistic and p-value. Here (http://knowledgetack.com/python/statsmodels/proportions_ztest/) is a helpful link on using the built in.

```
In [30]: # from - http://knowledgetack.com/python/statsmodels/proportions_ztest/

         z_score, p_value = sm.stats.proportions_ztest([convert_new,convert_old],
          [n_new,n_old],alternative='larger')
         print(z_score, p_value)
```

```
-1.31092419842 0.905058312759
```

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.**?

Put your answer here.

the z-score is 1.31092

the p-value is 0.90506

The p-value in part j was: 0.9026 so the two values are very close.

0.9026/0.9051 = .997 (Very close)

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?

Put your answer here.

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 [31]: df3=df2.copy()
         df3['intercept'] = 1
         df3.head()
         df3[['control','ab_page']] = pd.get_dummies(df3['group'])
         df3 = df3.drop('control', axis=1)
         df3.head()
```

Out[31]:

	user_id	timestamp	group	landing_page	converted	intercept	ab_page
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0	1	1
3	853541	2017-01-08 18:28:03.143765	treatment	new_page	0	1	1
6	679687	2017-01-19 03:26:46.940749	treatment	new_page	1	1	1
8	817355	2017-01-04 17:58:08.979471	treatment	new_page	1	1	1
9	839785	2017-01-15 18:11:06.610965	treatment	new_page	1	1	1

```
In [ ]:
```

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 [32]: log_mod = sm.Logit(df3['converted'], df3[['intercept', 'ab_page']])
         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.

```
In [33]: results.summary()
```

```
Out[33]:
```

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290584
Model:	Logit	**Df Residuals:**	290582
Method:	MLE	**Df Model:**	1
Date:	Mon, 27 Aug 2018	**Pseudo R-squ.:**	8.077e-06
Time:	16:55:34	**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

```
In [34]: np.exp(results.params)
```

```
Out[34]: intercept    0.136863
         ab_page      0.985123
         dtype: float64
```

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 the **Part II**?

Put your answer here.

The p-value associated with the ab_page is 0.190 it is based on two-tailed results 0.190/2 = 0.095 which matches up with the results in the calculations above. (http://www-01.ibm.com/support/docview.wss?uid=swg21482771 (http://www-01.ibm.com/support/docview.wss?uid=swg21482771) - Can one get one-tailed tests in Logistic Regression by dividing significance levels in half?)

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?

Put your answer here.

There are a lot of factors that might influence conversion beyond just the landing page. Example, what if the new landing page looks great on firefox, but not on chrome and it was designed on firefox. It could appear that the conversions are down for the new landing page, but no one checked which browser the user was using and it turned out the lack of conversions was simply a bug related to the browser.

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 (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.join.html)](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.join.html) 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 [35]:
```
countries_df = pd.read_csv('./data/countries.csv')
df_new = countries_df.set_index('user_id').join(df2.set_index('user_id'
), how='inner')

print(df_new['country'].unique())
df_new.head()
```

```
['UK' 'US' 'CA']
```

Out[35]:

	country	timestamp	group	landing_page	converted
user_id					
834778	UK	2017-01-14 23:08:43.304998	control	old_page	0
928468	US	2017-01-23 14:44:16.387854	treatment	new_page	0
822059	UK	2017-01-16 14:04:14.719771	treatment	new_page	1
711597	UK	2017-01-22 03:14:24.763511	control	old_page	0
710616	UK	2017-01-16 13:14:44.000513	treatment	new_page	0

In [36]:
```
### Create the necessary dummy variables

df_new['intercept'] = 1
df_new[['control','ab_page']] = pd.get_dummies(df_new['group'])
df_new = df_new.drop('control', axis=1)
df_new.head()
df_new[['CA','UK','US']] = pd.get_dummies(df_new['country'])
```

```
In [37]: df_new.head()
```

Out[37]:

	country	timestamp	group	landing_page	converted	intercept	ab_page
user_id							
834778	UK	2017-01-14 23:08:43.304998	control	old_page	0	1	0
928468	US	2017-01-23 14:44:16.387854	treatment	new_page	0	1	1
822059	UK	2017-01-16 14:04:14.719771	treatment	new_page	1	1	1
711597	UK	2017-01-22 03:14:24.763511	control	old_page	0	1	0
710616	UK	2017-01-16 13:14:44.000513	treatment	new_page	0	1	1

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 [38]: ### Fit Your Linear Model And Obtain the Results
         #log_mod = sm.OLS(df_new['converted'], df_new[['intercept','CA','UK']])
          # Using US as baseline
         #results = log_mod.fit(maxiter=1000)
         #print("DONE")
```

```
In [39]: #results.summary()
```

```
In [40]: #print(np.exp(results.params))
```

In [41]:
```
#Per https://classroom.udacity.com/nanodegrees/nd002/parts/682048c9-4e1a
-4020-8a47-7eaf3e34f0fe/modules/e4508dac-d083-427b-be3d-63663aeada68/les
sons/49462f74-b030-4bb6-bf67-8281c9181404/concepts/4fafac62-34d5-4067-8b
20-741c468968bf
# adding an interaction to the model

df_countries=df_new.copy()
df_countries['UK_ind_ab_page']=df_countries['UK']*df_countries['ab_page'
]
df_countries['CA_ind_ab_page']=df_countries['CA']*df_countries['ab_page'
]
df_countries['US_ind_ab_page']=df_countries['US']*df_countries['ab_page'
]
df_countries.head()
```

Out[41]:

	country	timestamp	group	landing_page	converted	intercept	ab_page
user_id							
834778	UK	2017-01-14 23:08:43.304998	control	old_page	0	1	0
928468	US	2017-01-23 14:44:16.387854	treatment	new_page	0	1	1
822059	UK	2017-01-16 14:04:14.719771	treatment	new_page	1	1	1
711597	UK	2017-01-22 03:14:24.763511	control	old_page	0	1	0
710616	UK	2017-01-16 13:14:44.000513	treatment	new_page	0	1	1

In [42]:
```
log_mod_m = sm.Logit(df_countries['converted'], df_countries[['intercep
t','UK_ind_ab_page','US_ind_ab_page']])
results_m = log_mod_m.fit()
print("DONE")
```

```
Optimization terminated successfully.
         Current function value: 0.366117
         Iterations 6
DONE
```

```
In [43]: results_m.summary()
```

```
Out[43]:
```

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290584
Model:	Logit	**Df Residuals:**	290581
Method:	MLE	**Df Model:**	2
Date:	Mon, 27 Aug 2018	**Pseudo R-squ.:**	1.082e-05
Time:	16:55:36	**Log-Likelihood:**	-1.0639e+05
converged:	True	**LL-Null:**	-1.0639e+05
		LLR p-value:	0.3164

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
intercept	-1.9926	0.008	-252.910	0.000	-2.008	-1.977
UK_ind_ab_page	0.0112	0.018	0.626	0.532	-0.024	0.046
US_ind_ab_page	-0.0144	0.012	-1.155	0.248	-0.039	0.010

```
In [44]: #print("Intercept: " + str(1/np.exp(results_m.params[0])))
         print("UK_ind_ab_page: " + str(np.exp(results_m.params[1])))
         print("US_ind_ab_page: " + str(1/np.exp(results_m.params[2])))


         print("Based on the results there is not a strong correlation between co
         untry and conversion.")
```

```
UK_ind_ab_page: 1.01129205352
US_ind_ab_page: 1.01452957326
Based on the results there is not a strong correlation between country
and conversion.
```

Conclusions

Congratulations on completing the project!

Gather Submission Materials

Once you are satisfied with the status of your Notebook, you should save it in a format that will make it easy for others to read. You can use the **File -> Download as -> HTML (.html)** menu to save your notebook as an .html file. If you are working locally and get an error about "No module name", then open a terminal and try installing the missing module using `pip install <module_name>` (don't include the "<" or ">" or any words following a period in the module name).

You will submit both your original Notebook and an HTML or PDF copy of the Notebook for review. There is no need for you to include any data files with your submission. If you made reference to other websites, books, and other resources to help you in solving tasks in the project, make sure that you document them. It is recommended that you either add a "Resources" section in a Markdown cell at the end of the Notebook report, or you can include a `readme.txt` file documenting your sources.

Submit the Project

When you're ready, click on the "Submit Project" button to go to the project submission page. You can submit your files as a .zip archive or you can link to a GitHub repository containing your project files. If you go with GitHub, note that your submission will be a snapshot of the linked repository at time of submission. It is recommended that you keep each project in a separate repository to avoid any potential confusion: if a reviewer gets multiple folders representing multiple projects, there might be confusion regarding what project is to be evaluated.

It can take us up to a week to grade the project, but in most cases it is much faster. You will get an email once your submission has been reviewed. If you are having any problems submitting your project or wish to check on the status of your submission, please email us at dataanalyst-project@udacity.com. In the meantime, you should feel free to continue on with your learning journey by beginning the next module in the program.