Analyze_ab_test_results_notebook

April 4, 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)
```

In [ ]:

In [ ]:

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

In [3]:
```
df.describe(include='all')
```

Out[3]:

	user_id	timestamp	group	landing_page	\
count	294478.000000	294478	294478	294478	
unique	NaN	294478	2	2	
top	NaN	2017-01-05 03:33:19.612263	treatment	new_page	
freq	NaN	1	147276	147239	
mean	787974.124733	NaN	NaN	NaN	
std	91210.823776	NaN	NaN	NaN	
min	630000.000000	NaN	NaN	NaN	
25%	709032.250000	NaN	NaN	NaN	
50%	787933.500000	NaN	NaN	NaN	
75%	866911.750000	NaN	NaN	NaN	
max	945999.000000	NaN	NaN	NaN	

	converted
count	294478.000000
unique	NaN
top	NaN
freq	NaN
mean	0.119659
std	0.324563
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	1.000000

In [4]:
```
df.dtypes
```

```
Out[4]: user_id          int64
        timestamp       object
        group           object
        landing_page    object
        converted        int64
        dtype: object
```

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

```
In [5]: len(df)
```

```
Out[5]: 294478
```

c. The number of unique users in the dataset.

```
In [6]: len(df.user_id.unique())
```

```
Out[6]: 290584
```

d. The proportion of users converted.

```
In [7]: round(len(df[df['converted']==1].user_id.unique())/len(df.user_id.unique()),2)
```

```
Out[7]: 0.12
```

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

```
In [8]: df1=df[df['group']=='treatment']
        len(df1[df1['landing_page']!='new_page'])
```

```
Out[8]: 1965
```

```
In [9]: df1=df[df['group']!='treatment']
        len(df1[df1['landing_page']=='new_page'])
```

```
Out[9]: 1928
```

f. Do any of the rows have missing values?

```
In [10]: df[df.isnull().any(axis=1)]
```

```
Out[10]: Empty DataFrame
         Columns: [user_id, timestamp, group, landing_page, converted]
         Index: []
```

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 [11]: df1=df[df['group']=='treatment']
         df0=df1[df1['landing_page']=='new_page']
         df11=df[df['group']=='control']
         df01=df11[df11['landing_page']=='old_page']
         #df13=df[df['group']=='control']
         #df03=df1[df1['landing_page']!='old_page']
         df2=df0
         df2=df2.append(df01)
         #df2=df2.append(df13)
         len(df2)
```

```
Out[11]: 290585
```

```
In [12]: # Double Check all of the correct rows were removed - this should be 0
         df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].sh
```

```
Out[12]: 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 [13]: len(df2['user_id'].unique())
```

```
Out[13]: 290584
```

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

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

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

```
In [15]: df2['user_id'].value_counts()
```

```
Out[15]: 773192    2
         630732    1
         811737    1
         797392    1
         795345    1
         801490    1
         799443    1
         787157    1
         793302    1
         817882    1
         842446    1
         815835    1
         805596    1
         803549    1
```

```
809694    1
807647    1
895712    1
840399    1
836301    1
899810    1
834242    1
936604    1
934557    1
940702    1
938655    1
830144    1
828097    1
832195    1
838348    1
821956    1
         ..
734668    1
736717    1
730574    1
775632    1
771538    1
642451    1
773587    1
783828    1
785877    1
779734    1
781783    1
759256    1
726472    1
748999    1
746950    1
753093    1
751044    1
740803    1
738754    1
744897    1
742848    1
634271    1
632222    1
636316    1
630169    1
650647    1
648598    1
654741    1
652692    1
630836    1
Name: user_id, Length: 290584, dtype: int64
```

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

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

```
Out[16]:       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 [17]: df2=df2.drop_duplicates(subset='user_id', keep='first', inplace=False)
```

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

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

```
In [18]: len(df2[df2['converted']==1])/len(df2)
```

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

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

```
In [19]: df300=df2[df2['group']=='control']
         cont_conv=len(df300[df300['converted']==1])/len(df2[df2['group']=='control'])
         cont_conv
```

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

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

```
In [20]: df300=df2[df2['group']=='treatment']
         treat_conv=len(df300[df300['converted']==1])/len(df2[df2['group']=='treatment'])
         treat_conv
```

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

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

```
In [21]: len(df2[df2['landing_page']=='new_page'])/len(df2)
```

```
Out[21]: 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.

Your answer goes here.
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: Old page is as good as the new page H1: New page is better than the old page

`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 [22]: _=df2.converted.mean()
         _
```

```
Out[22]: 0.11959708724499628
```

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

```
In [23]: _=df2.converted.mean()
         _
```

```
Out[23]: 0.11959708724499628
```

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

```
In [24]: _ =len(df2[df2['group']=='treatment'])
```

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

```
In [25]: _ =len(df2[df2['group']=='control'])
```

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 [26]: new_page_converted=np.random.choice([0,1], p=[(1 - _), _], size=_)
```

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 [27]: old_page_converted=np.random.choice([0,1], p=[(1 - p_ ), p_], size=_ )
```

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

```
In [28]: new_page_converted.mean()-old_page_converted.mean()

Out[28]: -0.00017433533771978038
```

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**.

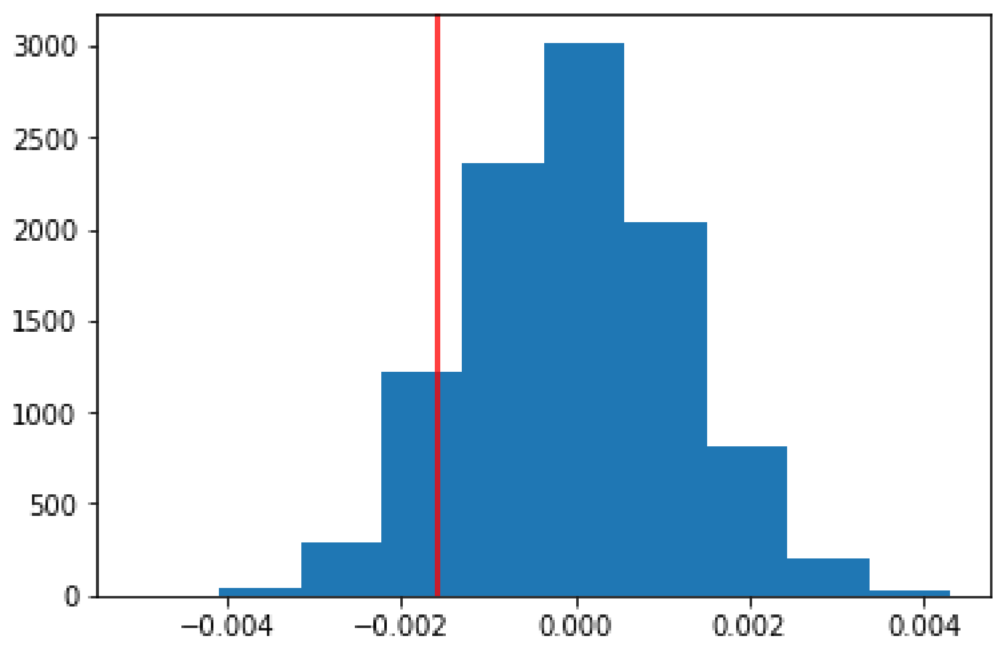
```
In [29]: p_diffs=[]
         for _ in range(10000):
             new_page_converted=np.random.choice([0,1], p=[(1 - _), _], size=_)
             old_page_converted=np.random.choice([0,1], p=[(1 - p_ ), p_], size=_ )
             new_page_converted.mean()-old_page_converted.mean()
             p_diffs.append(new_page_converted.mean()-old_page_converted.mean())

In [30]: p_diffs = np.asarray(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.

```
In [31]: act_diff=treat_conv-cont_conv
         print(act_diff)
         plt.hist(p_diffs)
         plt.axvline(x=act_diff, color='red');

-0.0015782389853555567
```



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

```
In [32]: (p_diffs>act_diff).mean()
```

```
Out[32]: 0.89470000000000005
```

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?

The calculated p-value (0.9013) is greater than 0.05 which is greater than α of 0.05. Therefore, we will fail to reject the null hypothesis.

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 [33]: import statsmodels.api as sm
         convert_old = df2.query('group == "control" & converted == 1')['converted'].count()
         convert_new = df2.query('group == "treatment" & converted == 1')['converted'].count()

         n_old = len(df2.query('group == "control"'))
         n_new = len(df2.query('group == "treatment"'))
```

```
/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 [34]: sm.stats.proportions_ztest([convert_new, convert_old], [n_new, n_old], alternative='sma
```

```
Out[34]: (-1.3109241984234394, 0.094941687240975514)
```

```
In [35]: from scipy.stats import norm

         # critical value:
         print(norm.ppf(1-(0.05)))
```

```
1.64485362695
```

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.**?

Since z-score is less than the critical value, we fail to reject the null hypothesis. This is consistent with the previous decision .

Part III - A regression approach

`1.` In this final part, you will see that the result you achieved in the A/B test in Part II above can also be achieved by performing regression.

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

Logistic Regression.

b. The goal is to use **statsmodels** to fit the regression model you specified in part **a.** to see if there is a significant difference in conversion based on which page a customer receives. However, you first need to create in df2 a column for the intercept, and create a dummy variable column for which page each user received. Add an **intercept** column, as well as an **ab_page** column, which is 1 when an individual receives the **treatment** and 0 if **control**.

In [39]:
```
pd.get_dummies(df2['group'])
df2[['old_page','ab_page']] = pd.get_dummies(df2['group'])
df2['intercept'] = 1
```

Out[39]:
```
        user_id                   timestamp    group landing_page  converted  \
294471   718310  2017-01-21 22:44:20.378320  control     old_page          0
294473   751197  2017-01-03 22:28:38.630509  control     old_page          0
294474   945152  2017-01-12 00:51:57.078372  control     old_page          0
294475   734608  2017-01-22 11:45:03.439544  control     old_page          0
294476   697314  2017-01-15 01:20:28.957438  control     old_page          0

        old_page  ab_page  intercept
294471         1        0          1
294473         1        0          1
294474         1        0          1
294475         1        0          1
294476         1        0          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 [45]:
```
log_mod = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])
result = log_mod.fit()
result.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:                Thu, 04 Apr 2019   Pseudo R-squ.:               8.077e-06
Time:                        01:26:56   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.

```
In [44]: result.summary()
```

```
Out[44]: <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, 04 Apr 2019   Pseudo R-squ.:               8.077e-06
Time:                        01:20:32   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
==============================================================================
"""
```

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 is 0.19. This is a p-value for two-tailed test. The corresponding p-value for one-tailed test will be 0.095 which is the same as the value obtained in part II. H_0 **:** p_{new} **-** p_{old} **= 0** H_1 **:** p_{new} **-** p_{old} **!= 0**

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?

A|B testing is prone to bias. Several factors could account for whether an individual converts or not. Demography could be among the factors.

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 [49]: countries=pd.read_csv("countries.csv")
         countries.head()
```

Out[49]:

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

```
In [50]: df2.head()
```

Out[50]:

	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	

	old_page	ab_page	intercept
2	0	1	1
3	0	1	1
6	0	1	1
8	0	1	1
9	0	1	1

```
In [52]: df2=df2.join(countries,lsuffix= "user_id", rsuffix="user_id")
```

```
In [55]: df2[['CA','UK', 'US']] = pd.get_dummies(df2['country'])
```

```
In [58]: log_mod = sm.Logit(df2['converted'], df2[['intercept', 'CA', 'UK']])
         result = log_mod.fit()
         result.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.366119
         Iterations 6
```

Out [58]:
```
<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, 04 Apr 2019   Pseudo R-squ.:               6.153e-06
Time:                        01:44:38   Log-Likelihood:            -1.0639e+05
converged:                       True   LL-Null:                   -1.0639e+05
                                        LLR p-value:                    0.5196
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -1.9966      0.007   -293.134      0.000      -2.010      -1.983
CA            -0.0258      0.027     -0.959      0.337      -0.079       0.027
UK             0.0065      0.013      0.490      0.624      -0.020       0.033
==============================================================================
"""
```

The p-values of the countries are greater than the critical values. So, once again, we fail to reject the null hypothesis

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 [61]:
```
df2['CA_page'] = df2['CA'] * df2['ab_page']
df2['UK_page'] = df2['UK'] * df2['ab_page']
df2.head()
```

Out [61]:
```
   user_iduser_id                   timestamp      group landing_page  \
2          661590  2017-01-11 16:55:06.154213  treatment     new_page
3          853541  2017-01-08 18:28:03.143765  treatment     new_page
6          679687  2017-01-19 03:26:46.940749  treatment     new_page
8          817355  2017-01-04 17:58:08.979471  treatment     new_page
9          839785  2017-01-15 18:11:06.610965  treatment     new_page

   converted  old_page  ab_page  intercept  user_iduser_id country  CA  UK  \
2          0         0        1          1        822059.0      UK   0   1
3          0         0        1          1        711597.0      UK   0   1
6          1         0        1          1        811617.0      US   0   0
```

```
8           1          0         1           1         887018.0       US    0    0
9           1          0         1           1         820683.0       US    0    0

   US  CA_page  UK_page
2   0        0        1
3   0        0        1
6   1        0        0
8   1        0        0
9   1        0        0
```

In [62]:
```
log_mod = sm.Logit(df2['converted'], df2[['intercept', 'CA', 'UK', 'CA_page', 'UK_page'
result = log_mod.fit()
result.summary()
```

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

Out[62]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              converted   No. Observations:               290584
Model:                          Logit   Df Residuals:                   290579
Method:                           MLE   Df Model:                            4
Date:                Thu, 04 Apr 2019   Pseudo R-squ.:               1.369e-05
Time:                        01:49:19   Log-Likelihood:            -1.0639e+05
converged:                       True   LL-Null:                   -1.0639e+05
                                        LLR p-value:                    0.5726
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -1.9966      0.007   -293.134      0.000      -2.010      -1.983
CA            -0.0568      0.038     -1.497      0.134      -0.131       0.018
UK             0.0120      0.018      0.684      0.494      -0.022       0.046
CA_page        0.0610      0.052      1.172      0.241      -0.041       0.163
UK_page       -0.0110      0.023     -0.479      0.632      -0.056       0.034
==============================================================================
"""
```

The p-values for all the variables are still higher than the critical value. Therefore, we fail to rejec the null hypothesis.

Finishing Up

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

Tip: Once you are satisfied with your work here, check over your report to make sure that it is satisfies all the areas of the rubric (found on the project submission page at the end of the lesson). You should also probably remove all of the "Tips" like this one so that the presentation is as polished as possible.

0.3 Directions to Submit

Before you submit your project, you need to create a .html or .pdf version of this notebook in the workspace here. To do that, run the code cell below. If it worked correctly, you should get a return code of 0, and you should see the generated .html file in the workspace directory (click on the orange Jupyter icon in the upper left).

Alternatively, you can download this report as .html via the **File** > **Download as** submenu, and then manually upload it into the workspace directory by clicking on the orange Jupyter icon in the upper left, then using the Upload button.

Once you've done this, you can submit your project by clicking on the "Submit Project" button in the lower right here. This will create and submit a zip file with this .ipynb doc and the .html or .pdf version you created. Congratulations!

```
In [65]: from subprocess import call
         call(['python', '-m', 'nbconvert', 'Analyze_ab_test_results_notebook.ipynb'])

Out[65]: 0
```