Analyze_ab_test_results_notebook

June 25, 2019

1 Analyze A/B Test Results

Mehdi Iddar Completed: May 25, 2019

1.1 Table of Contents

Introduction

A/B tests are very commonly performed by data analysts and data scientists.

For this project, I work to understand the results of a hypothetical A/B test run by an e-commerce website. The goal is to determine whether the company should implement the new page, keep the old page, or gather more information by running the experiment longer.

Part I - Probability

To get started, let's import our libraries.

```
In [2]: import pandas as pd
        import numpy as np
        import random
        import matplotlib.pyplot as plt
        %matplotlib inline
        random.seed(42)
```

1. reading in the `ab_data.csv` data. And Storing it in `df` as a DataFrame.

a. Reading in the data set and showing five first lines

```
In [3]: df = pd.read_csv('ab_data.csv')
        df.head()
```

Out[3]:

	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. Number of rows in the dataset :

In [4]:
```
# number of rows in the dataset
df.shape[0]
```

Out[4]: 294478

c. Number of unique users in the dataset :

In [5]:
```
# number of unique users
df['user_id'].nunique()
```

Out[5]: 290584

d. The proportion of users converted.

In [6]:
```
# the prop of users converted is the number of converted users divided by the total numb
prop_convert = df.query(" converted == 1")['converted'].count()/df.shape[0]
prop_convert
```

Out[6]: 0.11965919355605512

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

In [7]:
```
# There are two cases of non match :
# first : landing page is not new_page but group is treatment group
no_match_1 = df.query(" landing_page != 'new_page' and group == 'treatment'")['user_id']
# second : landing page is new_page but group is not treatment group
no_match_2 = df.query(" landing_page == 'new_page' and group != 'treatment'")['user_id']
# np_match is the sum of the two cases
no_match = no_match_1 + no_match_2
no_match
```

Out[7]: 3893

f. Checking if the rows have missing values

In [8]:
```
df.isnull().sum() # no messing values
```

Out[8]:
```
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.

a. Creating a new dataset that meets the specifications and storing the new dataframe in **df2**.

```
In [9]: # df_treatment is a dataframe where new_page and treatment match
        df_treatment = df.query("landing_page == 'new_page' and group == 'treatment' ")
```

```
In [10]: # df_control is a dataframe wherer old_page and control match
         df_control = df.query("landing_page == 'old_page' and group == 'control' ")
```

```
In [11]: # merging the two dataframes
         frames = [df_control, df_treatment]
         df2 = pd.concat(frames)
```

```
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. Checking the new dataframe df2 characteristics.

a. Number of unique **user_id**s in **df2**?

```
In [13]: # unique user_ids in df2
         df2['user_id'].nunique()
```

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

b. Checking repeated **user_id** in **df2**.

```
In [14]: # user_id : 773192 is repeated
         df2[df2.duplicated('user_id')]
```

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

c. Checking out the repeated **user_id**?

```
In [15]: # check out the repeated user_id
         df2.query("user_id == '773192' ")
```

```
Out[15]:       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. Removing **one** of the rows with a duplicate **user_id**

```
In [16]: # removing one of the repeated user_id
         df2 = df2.drop([1899])
```

4. Other Probabilities based on the df2 dataframe

a. The probability of an individual converting regardless of the page they receive

In [17]:
```
# the probability of coverting is the number of observations with converting divided by
df2_convert = df2.query(" converted == 1")
p_convert = df2.query(" converted == 1")['user_id'].count()/df2.shape[0]

p_convert
```

Out[17]: 0.11959708724499628

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

In [18]:
```
# let's create a subset of data corresponding to the control group
df2_control = df2.query(" group == 'control'")
```

In [19]:
```
# Calculate the probability of convert within the control group
p_control_convert = df2_control.query(" converted == 1")['user_id'].count()/df2_control

p_control_convert
```

Out[19]: 0.1203863045004612

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

In [20]:
```
# let's create a subset of data corresponding to the treatment group
df2_treatment = df2.query(" group == 'treatment'")
```

In [21]:
```
# Calculate the probability of convert within the treatment group
p_treatment_convert = df2_treatment.query(" converted == 1")['user_id'].count()/df2_tre

p_treatment_convert
```

Out[21]: 0.11880806551510564

d. The probability that an individual received the new page

In [22]:
```
# the probability of receiving the new page
p_newpage = df2.query(" landing_page == 'new_page'")['user_id'].count()/df2.shape[0]

p_newpage
```

Out[22]: 0.50006194422266881

In this sample, based on the results above the average convert within the treatment group (0.1188) is less than the average convert within the control group (0.1203). However we don't have the evidence that this result is statistacally significant and not due only to chance. We should go further with the analysis using hypothesis tests to make conclusion about whether or not the old page leads to more conversion

Part II - A/B Test

Because there is a time stamp associated with each user event, it is theoretically possible that a hypothesis test, similar to this one, could be run continuously in real time.

In this hypothetical situation, the hard questions become when to stop the test and how to reach a decision: As soon as one page is considered significantly better than the other? Once a certain page is better than another for a certain period of time? How long must the experiment be run before considering the test inconclusive? How much evidence is sufficient?

These are the difficult questions associated with A/B tests in general.

For the hypothetical A/B test considered in this section, the null and alternative hypotheses are as follows:

H0 : p_{new} - p_{old} <= 0 **H1** : p_{new} - p_{old} > 0

Or, verbally:

H0 : The likelihood of conversion for a user receiving the new page is less than or equal to the likelihood of conversion for a user receiving the old page. H1 : The likelihood of conversion for a user receiving the new page is greater than the likelihood of conversion for a user receiving the old page.

2. For the next section, assume the null hypothesis. That is, assume pnew and pold both have "true" success rates equal to the converted success rate regardless of 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 [23]: # conversion rate for     under the null
         p_new_H0 = df2['converted'].mean()
         p_new_H0

Out[23]: 0.11959708724499628
```

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

```
In [24]: # conversion rate for     under the null
         p_old_H0 = df2['converted'].mean()
         p_old_H0

Out[24]: 0.11959708724499628
```

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

```
In [25]: # number of individuals in the treatment group
         n_new = df2.query(" group == 'treatment'").shape[0]
         n_new

Out[25]: 145310
```

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

```
In [26]: # number of individuals in the control group
         n_old = df2.query(" group == 'control'").shape[0]
         n_old

Out[26]: 145274
```

Next, performing the sampling distribution for the difference in converted between the two pages over 10,000 iterations of calculating an estimate from the null.

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

```
In [27]: new_page_converted = np.random.binomial(1, p_new_H0, n_new )
         p_new = new_page_converted.mean()
```

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

```
In [28]: old_page_converted = np.random.binomial(1, p_old_H0, n_old )
         p_old = old_page_converted.mean()
```

g. What is p_{new} - p_{old} for our simulated values from part (e) and (f).

```
In [29]: diff = p_new - p_old
         diff

Out[29]: -0.0011998828064638811
```

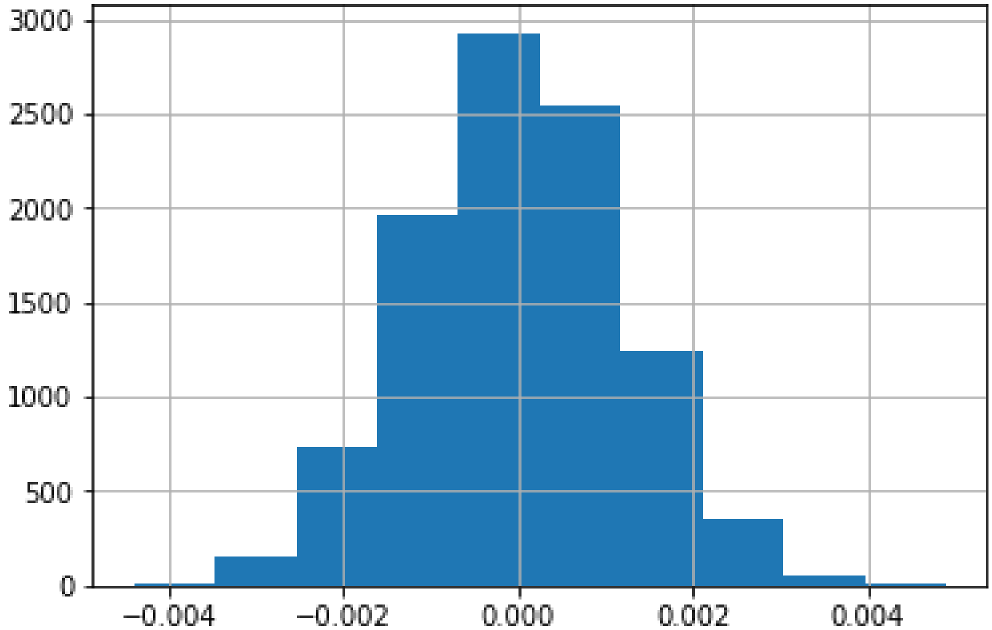
h. Creating 10,000 p_{new} - p_{old} values using the same simulation process I used in parts (a) through (g) above. Storing all 10,000 values in a NumPy array called **p_diffs**.

```
In [30]: p_diffs = []
         for _ in range(10000):
             new_page_converted = np.random.binomial(1, p_new_H0, n_new )
             old_page_converted = np.random.binomial(1, p_old_H0, n_old )
             p_new = new_page_converted.mean()
             p_old = old_page_converted.mean()
             p_diffs.append( p_new - p_old )
```

i. This boils down to a computation of the "spread" of the data, assuming that the probability of converting a given user is the same whether they see the treatment page or the control page.

```
In [31]: # lets transform p_diffs to a NumPy array
         p_diffs = np.array(p_diffs)
```

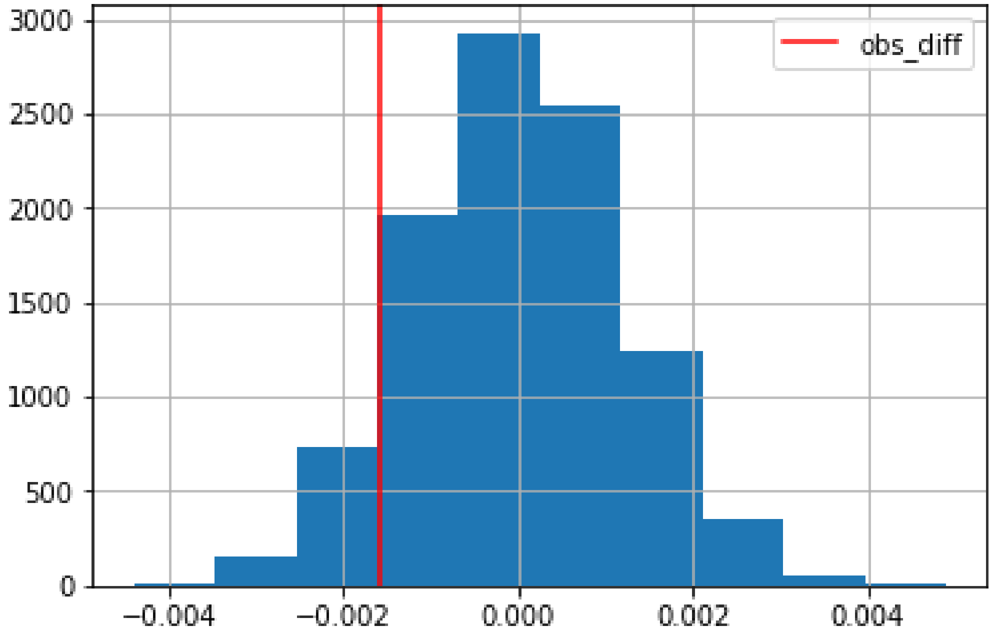
```
In [32]: # Plotting the histogram
         plt.hist(p_diffs);
         plt.grid()
```



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

In [33]:
```
# camputing the observed difference in ab_data.csv
obs_diff = p_treatment_convert - p_control_convert
```

In [34]:
```
# plotting both : histogram of p_diffs and the observed difference in ab_data.csv
plt.hist(p_diffs)
plt.grid()
plt.axvline(x = obs_diff, color='r', label='obs_diff')
plt.legend();
```




```
In [35]: # calculating the p_value
         p_value = (p_diffs > obs_diff).mean()
         p_value
```

```
Out[35]: 0.9062000000000001
```

We have calculated the p-value wish refers to the probability of observing our statistic or one more extreme in favor of the alternative given the null hypothesis is true. In our case the p-value (90%) is very large than the error type I rate (5%) we failed to reject the null hypothesis H0. we have sufficient evidence to conclude, there is no need to launch the new page given the old page leads to more conversions.

1.2 Built-In Stats Model

I 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.

Let's calculate the number of conversions for each page, as well as the number of individuals who received each page. `n_old` and `n_new` refer the the number of rows associated with the old page and new pages, respectively.

```
In [36]: 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'").shape[0]
```

```
n_new = df2.query(" group == 'treatment'").shape[0]
convert_old, convert_new, n_old, n_new
```

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

```
Out[36]: (17489, 17264, 145274, 145310)
```

m. Let's use `stats.proportions_ztest` to compute the test statistic and p-value.

```
In [37]: count = np.array([convert_old, convert_new])
         nobs = np.array([n_old, n_new])
         stat, pval = sm.stats.proportions_ztest(count, nobs, alternative='smaller')
         stat, pval
```

```
Out[37]: (1.3109241984234394, 0.90505831275902449)
```

The z-score is the number of standard deviations from the mean a data point. Since the z-score in this case of 1.31 is within the range implied by the critical value of 1.96, we fail to reject the null hypothesis.

On the other hand, since the p_value of 0.90 (approximately the same value as calculated before) is larger than the type I error rate (5%), we fail to reject the null hypothesis. We conclude that the two methods leads to the same conclusion.

Part III - A regression approach

`1.` This final part demonstrates that the result acheived in the previous A/B test can also be acheived by performing regression.

Since the response variable is a **categorical variable** (each row is either a conversion or no conversion), **logistic regression** is the type of regression that should be performed in this situation, since .

In this section, I utilize **statsmodels** to fit a logistic regression model to see if there is a significant difference in conversion depending on which page a customer receives.

First, I add a column for the intercept, called intercept. Then I create a dummy variable column for which page each user received. That column is called ab_page. It is 1 when a user receives treatment and 0 if the user receives control.

```
In [38]: # Adding an intercept column to the dataframe
         df2['intercept'] = 1
```

```
In [39]: # 'landing_page' dummy variables
         df_group_dummy = pd.get_dummies(df['group'])
```

```
In [40]: # merging dataframes  : df2 and df_page_dummy
         df2 = df2.join(df_group_dummy)
```

```
In [41]: # renaming treatment column to ab_page
         df2 = df2.rename(columns={'treatment': 'ab_page'})
```

```
In [42]: # dropping control column
         df2 = df2.drop('control' , axis = 1)
```

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

Out[43]:
```
   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
4   864975  2017-01-21 01:52:26.210827  control     old_page          1
5   936923  2017-01-10 15:20:49.083499  control     old_page          0
7   719014  2017-01-17 01:48:29.539573  control     old_page          0

   intercept  ab_page
0          1        0
1          1        0
4          1        0
5          1        0
7          1        0
```

c. The following line instantiates the regression model using **statsmodels**

In [44]:
```
# define the logistic regression model
logit_mod = sm.Logit(df2['converted'] , df2[['intercept' , 'ab_page']])
```

fitting the model using the intercept and ab_page columns. The model predicts whether or not a user converts.

In [45]:
```
result = logit_mod.fit()
```

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

d. Providing the summary of the model below

In [46]:
```
# model summary
result.summary()
```

Out[46]:
```
<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:                Tue, 25 Jun 2019   Pseudo R-squ.:               8.077e-06
Time:                        12:01:07   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**?

The p-value associated with ab_page gived by the logistic regression model is 0.19. The p-value of part II calculated from the built-in ztest method was about 0.90.

In part 3, The null hypothesis associated with our logistic regression model is that there is no relationship between the response variable and explanatory variables. This means there is no relationship between which page is displayed and the conversion rate. The alternative hypothes is that there is a relationship.

However, The null hypothesis from part 2 is : the likelihood of conversion for a user receiving the new page is less than or equal to the likelihood of conversion for a user receiving the old page. The alternative hypothesis is that the likelihood of conversion for a user receiving the new page is greater than the likelihood of conversion for a user receiving the old page.

There is a large difference in the p-values comparing the results of part 2 and part 3 We can explain this difference by some factors like that part 2 hypothesized one of the pages would lead to more conversions than the other. This is different from the hypotheses of part 3, which merely predicted a difference of some sort.

1.3 Additional Factors

Including additional factors may make the model more predictive, yielding greater understanding. It may also result in business insights that would not have been evident in this simpler anlysis. For example, it would be possible to have different versions of the website for different locations. It is likely that people from different countries might have different tastes in website layout.

Possible disadvantages of additional factors include increased risk of human error, especially misinterpretation, as well as possibly obscuring the message the data is really trying to tell (decreasing the so-called signal-to-noise ratio).

Next, I add an additional factor for the country in which a user lives. I read in an additional file called countries.csv and merge it with the users dataframe.

For future reference, Here are the docs for joining tables.

Finally, I generate dummy variables for these country columns.

In [47]:
```python
# reading countries data frame
df_countries = pd.read_csv('countries.csv')
df_countries.head()
```

Out[47]:

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

```
In [48]: # merge dataframes on 'user_id' using inner join
         df3 = df_countries.set_index('user_id').join(df2.set_index('user_id') , on = 'user_id'

In [49]: df3.head()
```

```
Out[49]:         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        0
         928468           0          1        1
         822059           1          1        1
         711597           0          1        0
         710616           0          1        1
```

```
In [50]: # setting dumy variables for country
         df3[['CA', 'UK' , 'US']] = pd.get_dummies(df3['country'])
         df3.head()
```

```
Out[50]:         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  CA  UK  US
         user_id
         834778           0          1        0   0   1   0
         928468           0          1        1   0   0   1
         822059           1          1        1   0   1   0
         711597           0          1        0   0   1   0
         710616           0          1        1   0   1   0
```

```
In [51]: # Fitting logistic regression model (US is the baseline)
         logmod = sm.Logit(df3['converted'] , df3[['intercept' , 'ab_page' , 'CA' , 'UK']])
         result = logmod.fit()
         result.summary()
```

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

Out[51]: <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:                Tue, 25 Jun 2019   Pseudo R-squ.:               2.323e-05
Time:                        12:07:47   Log-Likelihood:            -1.0639e+05
converged:                       True   LL-Null:                   -1.0639e+05
                                        LLR p-value:                    0.1760
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -1.9893      0.009   -223.763      0.000      -2.007      -1.972
ab_page       -0.0149      0.011     -1.307      0.191      -0.037       0.007
CA            -0.0408      0.027     -1.516      0.130      -0.093       0.012
UK             0.0099      0.013      0.743      0.457      -0.016       0.036
==============================================================================
"""
```

It is necessary to exponentiate these coefficients since this is logistic regression.

In [54]:
```
# Calculate exponential of coefficients
CA_coef = np.exp(-0.0408)
UK_coef = np.exp(0.0099)
print (CA_coef)
print (UK_coef)
```

```
0.960021114972
1.00994916712
```

In [53]:
```
# let's calculate the reciprocal for CA_coef
1/CA_coef
```

Out[53]: 1.0416437559600236

The interpretation of the foregoing variables is counterintuitive. In this case, United States is the baseline since it was the one out of three variables that wasn't included in the regression. We would say that CA users are 0.96 times as likely (or 4% less likely) to convert as US users. Similarly, we would say that UK users are 1.01 times as likely (or 1% more likely) to convert as US users.

The effect is not statistically significant, given the fairly large P-values. Even if it were, it is not clear that such a small difference between the different countries would be practically significant.

1.3.1 Interaction Terms

h. Though I have now looked at the individual factors of country and page on conversion, I would now like to look at an interaction between page and country to see if there significant effects on conversion.

Let's Create the necessary additional columns, and fit the new model.

In [58]:
```
#Columns corresponding to interaction between page and countries
df3['CA_int'] = df3['ab_page'] * df3['CA']
df3['UK_int'] = df3['ab_page'] * df3['UK']
df3['US_int'] = df3['ab_page'] * df3['US']
df3.head()
```

Out[58]:
```
             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  CA  UK  US  CA_int  UK_int  US_int
user_id
834778           0          1        0   0   1   0       0       0       0
928468           0          1        1   0   0   1       0       0       1
822059           1          1        1   0   1   0       0       1       0
711597           0          1        0   0   1   0       0       0       0
710616           0          1        1   0   1   0       0       1       0
```

In [57]:
```
# Fitting logistic regression model
logmod = sm.Logit(df3['converted'] , df3[['intercept' , 'ab_page' , 'CA' , 'CA_int' , '
result = logmod.fit()
result.summary()
```

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

Out[57]:
```
<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:                Tue, 25 Jun 2019   Pseudo R-squ.:               3.482e-05
Time:                        12:13:59   Log-Likelihood:            -1.0639e+05
```

```
converged:                       True   LL-Null:                  -1.0639e+05
                                        LLR p-value:                   0.1920
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -1.9865      0.010   -206.344      0.000      -2.005      -1.968
ab_page       -0.0206      0.014     -1.505      0.132      -0.047       0.006
CA            -0.0175      0.038     -0.465      0.642      -0.091       0.056
CA_int        -0.0469      0.054     -0.872      0.383      -0.152       0.059
UK            -0.0057      0.019     -0.306      0.760      -0.043       0.031
UK_int         0.0314      0.027      1.181      0.238      -0.021       0.084
==============================================================================
"""
```

Giving the high p-values associated with this logistic regression for the case with the interaction terms, we can fairly conclude that the interaction terms are not especially predictive of results.

Conclusion

Throughout this project, we run two different aproches (a/b tests and logistic regresion) to study the results of implementation of a new page for an ecommerce website. Given the results of both approches the conclusion of this study has to be that the the new page does not have a statistically significant impact on new user conversions. This mean the company should stick with the old web page.