Analyze_ab_test_results

November 20, 2017

0.1 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!

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

```
In [1]: # import libraries
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        from tqdm import *

        %matplotlib inline
        #We are setting the seed to assure you get the same answers on quizzes as we set up
        np.random.seed(42)
```

`Question 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]: # import data
        df = pd.read_csv('ab_data.csv')

        # show top rows
        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]: # Calculate number of rows in dataset and display
        df_length = len(df)
        print(df_length)
```

```
294478
```

c. Number of unique users in the dataset.

```
In [4]: # Calculate number of unique users in dataset
        len(df.user_id.unique())
```

```
Out[4]: 290584
```

d. The proportion of users converted.

```
In [5]: df.converted.sum()/df_length
```

```
Out[5]: 0.11965919355605512
```

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

```
In [6]: # Looking for rows where treatment/control doesn't line up with old/new pages respecti
        df_t_not_n = df[(df['group'] == 'treatment') & (df['landing_page'] == 'old_page')]
        df_not_t_n = df[(df['group'] == 'control') & (df['landing_page'] == 'new_page')]

        # Add lengths
        mismatch= len(df_t_not_n) + len(df_not_t_n)

        # Create one dataframe from it
        mismatch_df = pd.concat([df_t_not_n, df_not_t_n])

        mismatch
```

```
Out[6]: 3893
```

f. Do any rows have missing values?

```
In [7]: # Check for missing values?
        df.isnull().values.any()
```

```
Out[7]: False
```

Question 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]: # Copy dataframe
        df2 = df

        # Remove incriminating rows
        mismatch_index = mismatch_df.index
        df2 = df2.drop(mismatch_index)
```

```
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].sl
```

```
Out[9]: 0
```

Question 3

a. How many unique user_ids are in df2?

```
In [10]: # Find unique users
         print("Unique users:", len(df2.user_id.unique()))

         # Check for not unique users
         print("Non-unique users:", len(df2)-len(df2.user_id.unique()))
```

```
Unique users: 290584
Non-unique users: 1
```

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

```
In [11]: # Find duplicated user
         df2[df2.duplicated('user_id')]
```

Out[11]:

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

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

```
In [12]: # Find duplicates under user ids
         df2[df2['user_id']==773192]
```

Out[12]:
```
      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, keep dataframe as df2

In [13]:
```
# Drop duplicated user
df2.drop(labels=1899, axis=0, inplace=True)
```

In [14]:
```
# Check the drop worked
df2[df2['user_id']==773192]
```

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

Question 4

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

In [15]:
```
# Probability of user converting
print("Probability of user converting:", df2.converted.mean())
```

```
Probability of user converting: 0.11959708724499628
```

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

In [16]:
```
# Probability of control group converting
print("Probability of control group converting:",
      df2[df2['group']=='control']['converted'].mean())
```

```
Probability of control group converting: 0.1203863045004612
```

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

In [17]:
```
# Probability of treatment group converting
print("Probability of treatment group converting:",
      df2[df2['group']=='treatment']['converted'].mean())
```

```
Probability of treatment group converting: 0.11880806551510564
```

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

In [18]:
```
# Probability an individual recieved new page
print("Probability an individual recieved new page:",
      df2['landing_page'].value_counts()[0]/len(df2))
```

```
Probability an individual recieved new page: 0.500061944223
```

`e.` Use the results in the previous two portions of this question to suggest if you think there is evidence that one page leads to more conversions? Write your response below.

Given the data in Question 4 so far, the probability that an individual recieved a new page is roughly 0.5, this means that it is not possible for there to be a difference in conversion based on being given more opportunities to do so. For instance, if the probability of recieving a new page was higher relative to the old page then it would be observed that the rate of conversion would naturally increase.

The probabilities that the control converted at higher rates seems to make sense in the data, however the magnitude of this change is very small(0.2%). It would be good to do a test to see whether or not it is statistically significant. Practically, I would say that the gain of the new site is negligible. I would want to see a larger increase (2-5%) before a decision is made for which site is better.

Part II - A/B Test

`Question 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.

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

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

`Question 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?

Under null hypothesis, $p_{new} \leq p_{old}$. Hence, we should calculate the average of the real p_{new} and p_{old} from the data and let this average value be the value we use. (Confusing sentence, hopefully you get what I mean.)

```
In [19]: # Calculate probability of conversion for new page
         p_new = df2[df2['landing_page']=='new_page']['converted'].mean()

         print("Probability of conversion for new page:", p_new)
```

```
Probability of conversion for new page: 0.11880806551510564
```

```
In [20]: # Calculate probability of conversion for old page
         p_old = df2[df2['landing_page']=='old_page']['converted'].mean()

         print("Probability of conversion for old page:", p_old)
```

```
Probability of conversion for old page: 0.1203863045004612
```

```
In [21]: # Take the mean of these two probabilities
         p_mean = np.mean([p_new, p_old])

         print("Probability of conversion udner null hypothesis:", p_mean)
```

```
Probability of conversion udner null hypothesis: 0.119597185008
```

```
In [22]: # Calc. differences in probability of conversion for new and old page (not under H_0)
         p_diff = p_new-p_old

         print("Difference in probability of conversion for new and old page (not under H_0):"
```

```
Difference in probability of conversion for new and old page (not under H_0): -0.00157823898535
```

Hence:
p_{new} : 0.1188
p_{old} : 0.1204
a. What is the **convert rate** for p_{new} under the null?
$p_{mean} = p_{old_0} = p_{new_0}$: 0.1196
b. What is the **convert rate** for p_{old} under the null?
The same thing.
$p_{new_0} - p_{old_0}$: 0

```
In [23]: # Calculate n_new and n_old
         n_new, n_old = df2['landing_page'].value_counts()

         print("new:", n_new, "\nold:", n_old)
```

```
new: 145310
old: 145274
```

Hence:
c. What is n_{new}?
n_{new} : 145310
d. What is n_{old}?
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 [24]: # Simulate conversion rates under null hypothesis
         new_page_converted = np.random.choice([1, 0], size=n_new, p=[p_mean, (1-p_mean)])

         new_page_converted.mean()
```

Out[24]: 0.11979216846741449

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 [25]:
```
# Simulate conversion rates under null hypothesis
old_page_converted = np.random.choice([1, 0], size=n_old, p=[p_mean, (1-p_mean)])

old_page_converted.mean()
```

Out[25]: 0.11925051970758704

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

In [26]:
```
# Calculate difference in p under the null hypothesis
new_page_converted.mean()-old_page_converted.mean()
```

Out[26]: 0.00054164875982744276

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 **p_diffs**.

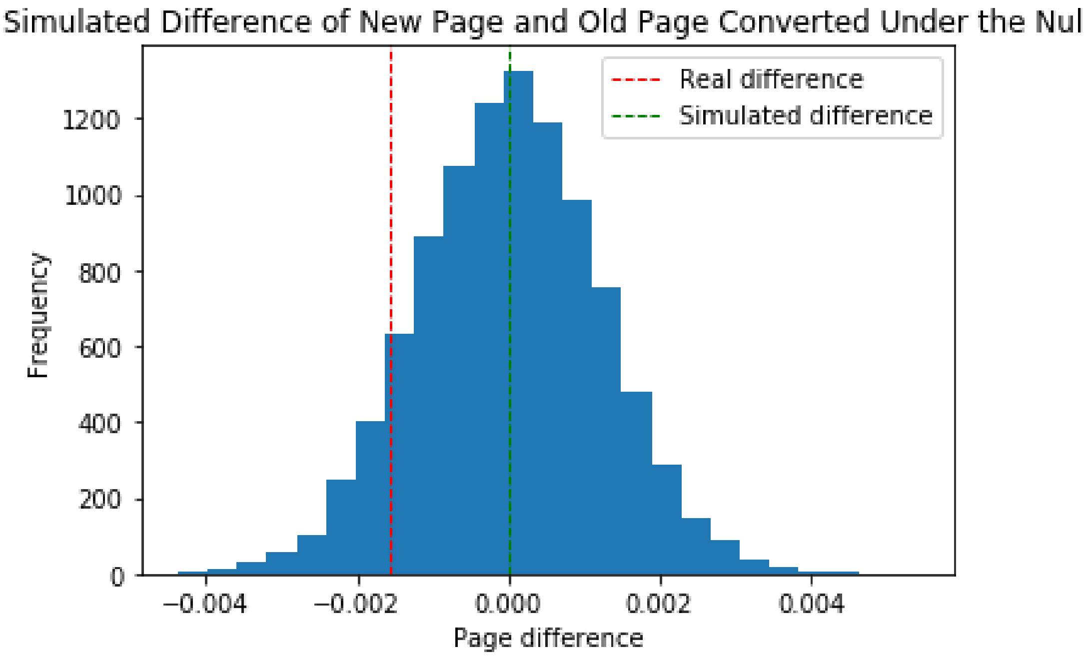
In [27]:
```
p_diffs = []

# Re-run simulation 10,000 times
# trange creates an estimate for how long this program will take to run
for i in trange(10000):
    new_page_converted = np.random.choice([1, 0], size=n_new, p=[p_mean, (1-p_mean)])
    old_page_converted = np.random.choice([1, 0], size=n_old, p=[p_mean, (1-p_mean)])
    p_diff = new_page_converted.mean()-old_page_converted.mean()
    p_diffs.append(p_diff)
```

```
100%|| 10000/10000 [00:40<00:00, 249.47it/s]
```

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 [28]:
```
# Plot histogram
plt.hist(p_diffs, bins=25)
plt.title('Simulated Difference of New Page and Old Page Converted Under the Null')
plt.xlabel('Page difference')
plt.ylabel('Frequency')
plt.axvline(x=(p_new-p_old), color='r', linestyle='dashed', linewidth=1, label="Real d
plt.axvline(x=(np.array(p_diffs).mean()), color='g', linestyle='dashed', linewidth=1,
plt.legend()
plt.show()
```



The simulated data creates a normal distribution (no skew) as expected (it was generated at random after all). However, the mean of this normal distribution is 0 which suggests that the old page has the same higher conversion rate than the new page on average. (Which is what the data should look like under the null hypothesis.)

Whether or not real value is different enough to be significant depends on where it falls on this bell curve

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

```
In [29]: # Find proportion of p_diffs greater than the actual difference
         greater_than_diff = [i for i in p_diffs if i > (p_new-p_old)]
```

```
In [30]: # Calculate values
         print("Actual difference:" , p_new-p_old)

         print('Proportion greater than actual difference:', len(greater_than_diff)/len(p_diffs
```

```
Actual difference: -0.0015782389853555567
Proportion greater than actual difference: 0.9065
```

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?

If our sample conformed to the null hypothesis then we'd expect the proportion greater than the actual difference to be 0.5. However, we calculate that almost 90% of the population in our simulated sample lies above the real difference which suggests that the real sample likely does not comform to the null hypothesis.

I'm not sure what this value is called however.

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 [31]:
```
# Import statsmodels
import statsmodels.api as sm

# Calculate number of conversions
convert_old = len(df2[(df2['landing_page']=='new_page')&(df2['converted']==1)])
convert_new = len(df2[(df2['landing_page']=='old_page')&(df2['converted']==1)])

print("convert_old:", convert_old,
      "\nconvert_new:", convert_new,
      "\nn_old:", n_old,
      "\nn_new:", n_new)
"""
Some of these values were defined ealier in this notebook: n_old and n_new
"""
```

```
convert_old: 17264
convert_new: 17489
n_old: 145274
n_new: 145310


/home/simon/anaconda3/lib/python3.6/site-packages/statsmodels/compat/pandas.py:56: FutureWarni
  from pandas.core import datetools
```

Out[31]:
```
'\nSome of these values were defined ealier in this notebook: n_old and n_new\n'
```

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 [32]:
```
# Find z-score and p-value
z_score, p_value = sm.stats.proportions_ztest(count=[convert_new, convert_old],
                                              nobs=[n_new, n_old])
print("z-score:", z_score,
     "\np-value:", p_value)
```

```
z-score: 1.26169574219
p-value: 0.207058289607
```

`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.**?

> Simply put, a z-score is the number of standard deviations from the mean a data point is. But more technically it's a measure of how many standard deviations below or above the population mean a raw score is

-Source

Given the above definition, it would seem that the differences between the lines shown in the histogram above is 1.26 standard deviations, which looks like a lot but is not satistically sifnificant. The p-value is roughly 20.7% which is the probability that this result is due to random chance which means that we can only really reject the null-hypothesis with a ~75% confidence. (Which you really wouldn't do under normal circumstances, so in reality we fail to reject the null hypothesis)

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?

You can use a logistic regression.

This will likely be the `sm` module to use.

`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 [33]: df3 = df2 # Clone dataframe in case of a mistake
```

```
In [34]: df3['intercept'] = pd.Series(np.zeros(len(df3)), index=df3.index)
         df3['ab_page'] = pd.Series(np.zeros(len(df3)), index=df3.index)
```

```
In [35]: # Find indexes that need to be changed for treatment group
         index_to_change = df3[df3['group']=='treatment'].index

         # Change values
         df3.set_value(index=index_to_change, col='ab_page', value=1)
         df3.set_value(index=df3.index, col='intercept', value=1)

         # Change datatype
         df3[['intercept', 'ab_page']] = df3[['intercept', 'ab_page']].astype(int)

         # Move "converted" to RHS
         df3 = df3[['user_id', 'timestamp', 'group', 'landing_page', 'ab_page', 'intercept', '
```

```
/home/simon/anaconda3/lib/python3.6/site-packages/ipykernel/__main__.py:5: FutureWarning: set_
/home/simon/anaconda3/lib/python3.6/site-packages/ipykernel/__main__.py:6: FutureWarning: set_
```

```
In [36]: # Check everything  has worked
         df3[df3['group']=='treatment'].head()
```

```
Out[36]:    user_id                   timestamp      group landing_page  ab_page  \
         2   661590  2017-01-11 16:55:06.154213  treatment     new_page        1
         3   853541  2017-01-08 18:28:03.143765  treatment     new_page        1
         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

            intercept  converted
         2          1          0
         3          1          0
         6          1          1
         8          1          1
         9          1          1
```

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 [37]: # Set up logistic regression
         logit = sm.Logit(df3['converted'], df3[['ab_page', 'intercept']])

         # Calculate results
         result=logit.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 [38]: result.summary2() # result.summary() wasn't working for some reason, but this one doe

Out[38]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                  Results: Logit
         ==================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2017-11-20 16:00 AIC:              212780.3502
         No. Observations:   290584           BIC:              212801.5095
         Df Model:           1                Log-Likelihood:   -1.0639e+05
         Df Residuals:       290582           LL-Null:          -1.0639e+05
         Converged:          1.0000           Scale:            1.0000
         -------------------------------------------------------------------
                       Coef.   Std.Err.      z      P>|z|     [0.025    0.975]
```

```
------------------------------------------------------------------------------
ab_page       -0.0150     0.0114     -1.3109  0.1899  -0.0374    0.0074
intercept     -1.9888     0.0081   -246.6690  0.0000  -2.0046   -1.9730
==============================================================================

"""
```

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

Apparently the p-value associated with `ab_page` is 0.1899, which is slightly lower than the p-value I calculated using the z-test above. The reason why the value is lower is because I added an intercept which is meant to account for error if my memory is correct. This means that this value is more accurate. (As in, it's probably closer to the true p-value)

However, this p-value is still much too high to reject the null hypothesis.

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?

There are certainly disadvantages to adding too many features into your analysis. When do you regression or categorization analysis you want to have features which have large impacts on outcome, small impacts are usually not influencial and should be left for the intercept.

I believe there's a statistic which accounts for this, some sort of corrected Rš value (in linear regression at least) which will give lower outputs if "useless" features are added.

However, only one feature was chosen to determine whether a user would convert (beside the intercept) so a couple of added features wouldn't hurt. I would imagine some features like the time spent looking at page and the date the page was designed might be some interesting features to add. The longer a customer spends on a page the more they are likely to be content with it and unwilling to change, it could also be the case that really old pages will not work well and people will want an updated version.

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

In [39]:
```python
# Importing data
df_countries = pd.read_csv('countries.csv')

df_countries.head()
```

Out[39]:

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

In [40]:
```
# Creating dummy variables
df_dummy = pd.get_dummies(data=df_countries, columns=['country'])

# Performing join
df4 = df_dummy.merge(df3, on='user_id') # df.join is depricated AFAIK

# Sorting columns
df4 = df4[['user_id', 'timestamp', 'group', 'landing_page',
           'ab_page', 'country_CA', 'country_UK', 'country_US',
           'intercept', 'converted']]

# Fix Data Types
df4[['ab_page', 'country_CA', 'country_UK', 'country_US','intercept', 'converted']] =\
df4[['ab_page', 'country_CA', 'country_UK', 'country_US','intercept', 'converted']].a

df4.head()
```

Out[40]:
```
   user_id                   timestamp      group landing_page  ab_page  \
0   834778  2017-01-14 23:08:43.304998    control     old_page        0
1   928468  2017-01-23 14:44:16.387854  treatment     new_page        1
2   822059  2017-01-16 14:04:14.719771  treatment     new_page        1
3   711597  2017-01-22 03:14:24.763511    control     old_page        0
4   710616  2017-01-16 13:14:44.000513  treatment     new_page        1

   country_CA  country_UK  country_US  intercept  converted
0           0           1           0          1          0
1           0           0           1          1          0
2           0           1           0          1          1
3           0           1           0          1          0
4           0           1           0          1          0
```

In [41]:
```
# Create logit_countries object
logit_countries = sm.Logit(df4['converted'],
                           df4[['country_UK', 'country_US', 'intercept']])

# Fit
result2 = logit_countries.fit()
```

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

In [42]:
```
# Show results
result2.summary2()
```

Out[42]:
```
<class 'statsmodels.iolib.summary2.Summary'>
"""
                              Results: Logit
```

```
==================================================================
Model:              Logit            No. Iterations:   6.0000
Dependent Variable: converted        Pseudo R-squared: 0.000
Date:               2017-11-20 16:00 AIC:              212780.8333
No. Observations:   290584           BIC:              212812.5723
Df Model:           2                Log-Likelihood:   -1.0639e+05
Df Residuals:       290581           LL-Null:          -1.0639e+05
Converged:          1.0000           Scale:            1.0000
-------------------------------------------------------------------
               Coef.   Std.Err.     z      P>|z|    [0.025   0.975]
-------------------------------------------------------------------
country_UK     0.0507    0.0284    1.7863  0.0740  -0.0049   0.1064
country_US     0.0408    0.0269    1.5178  0.1291  -0.0119   0.0935
intercept     -2.0375    0.0260  -78.3639  0.0000  -2.0885  -1.9866
==================================================================

"""
```

It seems that country did have some bearing on conversion rate, but not high enough to be satistically significant

`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 are 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 [43]:
```
# Create logit_countries object
logit_countries2 = sm.Logit(df4['converted'],
                           df4[['ab_page', 'country_UK', 'country_US', 'intercept']])

# Fit
result3 = logit_countries2.fit()
```

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

In [44]:
```
# Show results
result3.summary2()
```

Out[44]:
```
<class 'statsmodels.iolib.summary2.Summary'>
"""
                          Results: Logit
==================================================================
Model:              Logit            No. Iterations:   6.0000
Dependent Variable: converted        Pseudo R-squared: 0.000
Date:               2017-11-20 16:00 AIC:              212781.1253
No. Observations:   290584           BIC:              212823.4439
Df Model:           3                Log-Likelihood:   -1.0639e+05
```

```
Df Residuals:        290580          LL-Null:           -1.0639e+05
Converged:           1.0000          Scale:             1.0000
-------------------------------------------------------------------
               Coef.   Std.Err.     z      P>|z|    [0.025   0.975]
-------------------------------------------------------------------
ab_page       -0.0149    0.0114   -1.3069  0.1912  -0.0374   0.0075
country_UK     0.0506    0.0284    1.7835  0.0745  -0.0050   0.1063
country_US     0.0408    0.0269    1.5161  0.1295  -0.0119   0.0934
intercept     -2.0300    0.0266  -76.2488  0.0000  -2.0822  -1.9778
==================================================================

"""
```

When adding everything together it seems that the p-values for all featues has increased. The z-score for the intercept is incredibly large though which is interesting.

Conclusions

Although it would seem from the outset that there is a difference between the conversion rates of new and old pages, there is just not enough evidence to reject the null hypothesis. That is, the difference in the conversion rates between new and old pages are not just not great enough to say with at least 95% certainity that it's just a random variation. (Especially because multiple methods were used to analyze the data.)

It was also found that this was not dependent on countries with conversion rates being roughly the same in the UK as in the US. The test conditions were fairly good as well, users had a roughly 50% chance to recieve the new and old pages and the sample size of the initial dataframe is sufficiently big such that collecting data is likely not a good use of resources.

I would recommend that the e-commerce company spend their money on trying to improve their website before trying again.