Analyze_ab_test_results_notebook

July 7, 2018

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

- Section **??**
- Section **??**
- Section **??**
- Section **??**

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 [3]: import pandas as pd
        import numpy as np
        import random
        import matplotlib.pyplot as plt
        %matplotlib inline
        #We are setting the seed to assure you get the same answers on quizzes as we set up
        random.seed(42)
```

1. Now, read in the `ab_data.csv` data. Store it in `df`. **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 [4]: # Import Data and Read CSV file
        df = pd.read_csv('ab_data.csv')

        # Initial inspection
        df.head()
```

Out[4]:

	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 [5]: # Use Shape Function To get a count of rows
        df.shape[0]
```

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

c. The number of unique users in the dataset.

```
In [6]: # Number of Unique Users

        df.user_id.nunique()
```

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

d. The proportion of users converted.

```
In [7]: df.converted.mean()

        # Conversion Rate Approx ~12%
```

```
Out[7]: 0.11965919355605512
```

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

```
In [8]: # Instances of treatmeant group lands on old page
        treatment_old_page = df.query("group == 'treatment' and landing_page == 'old_page'")
        print ("Treatment group lands on old_page {} times".format(len(treatment_old_page)))

        # Instances of Control Landing on new_page
        control_new_page = df.query("group == 'control' and landing_page == 'new_page'")
        print ("Control group lands on new_page {} times".format(len(control_new_page)))

        # Number of times New_page and Treatment are Misaligned Total
        total_mismatch = df.query('(group == "treatment") != (landing_page == "new_page")').user
        print ("The total number of mismatches is {}".format(total_mismatch))
```

```
Treatment group lands on old_page 1965 times
Control group lands on new_page 1928 times
The total number of mismatches is 3893
```

```
In [9]: #Number of times they ARE aligned Correctly
        df.query('group == "treatment" and landing_page == "new_page"').user_id.count()

Out[9]: 145311
```

f. Do any of the rows have missing values?

```
In [10]: #Check For missing values
         df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 294478 entries, 0 to 294477
Data columns (total 5 columns):
user_id         294478 non-null int64
timestamp       294478 non-null object
group           294478 non-null object
landing_page    294478 non-null object
converted       294478 non-null int64
dtypes: int64(2), object(3)
memory usage: 11.2+ MB
```

No Missing Values

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 [11]: mismatch = df.query('(group == "treatment") != (landing_page == "new_page")')
         df2 = df.drop(mismatch.index)

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.

```
In [13]: # Double Check
         df2.head()
```

```
Out[13]:    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 [14]: df.tail()
```

```
Out[14]:         user_id                   timestamp      group landing_page  converted
         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
         294477   715931  2017-01-16 12:40:24.467417  treatment     new_page          0
```

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

```
In [15]: df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 290585 entries, 0 to 294477
Data columns (total 5 columns):
user_id         290585 non-null int64
timestamp       290585 non-null object
group           290585 non-null object
landing_page    290585 non-null object
converted       290585 non-null int64
dtypes: int64(2), object(3)
memory usage: 13.3+ MB
```

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

```
In [16]: df2.nunique()['user_id']
```

```
Out[16]: 290584
```

```
In [17]: # Check number of duplicates
         sum(df2['user_id'].duplicated())
```

```
Out[17]: 1
```

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

```
In [18]: # Use duplicated function to view duplicate ID

         df2[df2.duplicated(['user_id'], keep=False)]['user_id']
```

```
Out[18]: 1899    773192
         2893    773192
         Name: user_id, dtype: int64
```

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

```
In [19]: # Drop duplicates with drop_duplicates fn
         df2 = df2.drop_duplicates(['user_id'])
```

```
In [20]: # Check if drop was successful
         sum(df2['user_id'].duplicated())
```

```
Out[20]: 0
```

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 [21]: df2.converted.mean()
```

```
Out[21]: 0.11959708724499628
```

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

```
In [22]: # Get control group probability
         control_conversion = df2[df2['group'] == 'control']['converted'].mean()
         print(control_conversion)

         # Round to 5 decimal places
         control_rounded = round(control_conversion, 4)
         print(control_rounded)
```

```
0.1203863045
0.1204
```

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

```
In [23]: # Get treatment group probability
         treatment_conversion = df2[df2['group'] == 'treatment']['converted'].mean()
         print(treatment_conversion)

         #round to 5 decimal places
         treatment_rounded =round(treatment_conversion, 4)
         print(treatment_rounded)
```

```
0.118808065515
0.1188
```

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

```
In [24]: # Probability of of recieving new page
         df2['landing_page'].value_counts()[0] / len(df2)

Out[24]: 0.50006194422266881

In [25]: # Calculate difference of conversion
         conversion_difference = control_rounded - treatment_rounded
         print(conversion_difference)

0.0016
```

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.

If we consider that the overall probability, meaning the control and treatment group combined, is ~12% overall and that the probabilities for the control and treatment group are 12.04% and 11.88 respectively, there is not sufficient evidence to suggest that the new treatment page leads to significantly more conversions. The difference between the two being about 0.15% and roughly the same distance from the overall conversion rate of 11.88%. Furthermore, the probability of recieving a new page and old page was 50%.

Because the conversion percentages are so similar it would be prudent to collect more data to see if there was any variation in test timing, change aversion or other factors that influenced the conversion rate for the new and old pages before drawing any concrete conclusions.

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.

Hypothesis

$H_0 : p_{old} >= p_{new}$ $H_1 : p_{old} < p_{new}$

Or stated Differently:

$H_0 : p_{new} <= p_{old}$ $H_1 : p_{new} > p_{old}$

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 [26]: # convert rate of P under new null
         p_new = df2.converted.mean()
         p_new

Out[26]: 0.11959708724499628
```

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

```
In [27]: # convert rate of p_old under the null
         p_old = df2.converted.mean()
         p_old

Out[27]: 0.11959708724499628
```

c. What is n_{new}?

```
In [28]: n_new = df2.query('group == "treatment"').shape[0]
         n_new

Out[28]: 145310
```

d. What is n_{old}?

```
In [29]: #shape of old control group

         n_old = df2.query('group == "control"').shape[0]
         n_old

Out[29]: 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 [30]: new_page_converted = np.random.binomial(n_new, p_new)
         new_page_converted

Out[30]: 17217
```

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 [31]: old_page_converted = np.random.binomial(n_old, p_old)
         old_page_converted
```

```
Out[31]: 17210
```

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

```
In [32]: (new_page_converted/n_new) - (old_page_converted/n_old)
```

```
Out[32]: 1.8823421346916835e-05
```

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**.

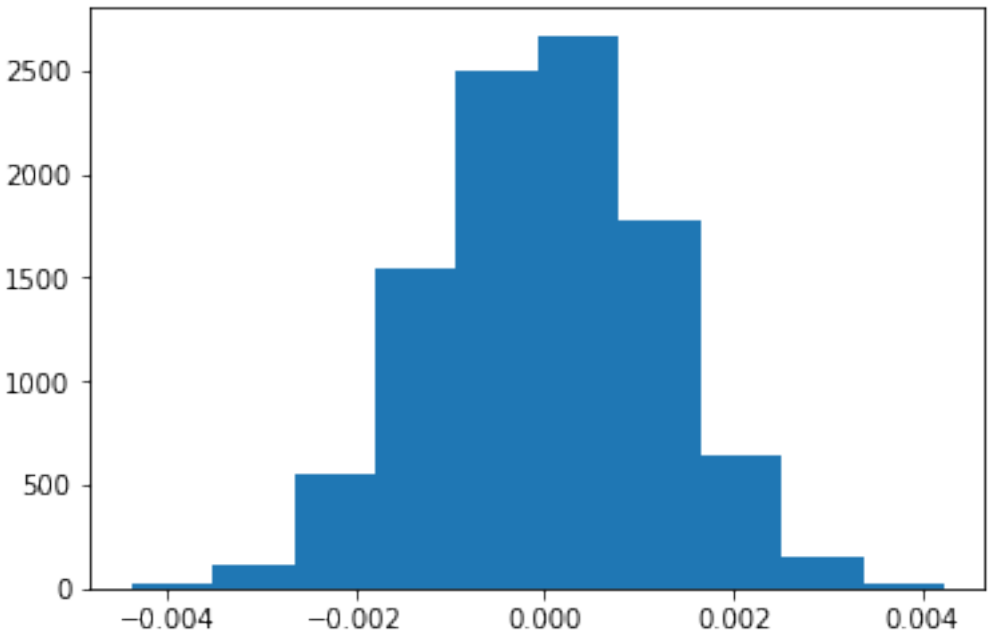
```
In [34]: # Because the shape value for n_new and n_old are different ther will any comparison wi
         # To circumvent this error we will use the mean. This should not change our output sinc
         # probabilities as in the previous case.

         p_diffs = np.random.binomial(n_new, p_new, 10000)/n_new - np.random.binomial(n_old, p_n
```

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 [35]: plt.hist(p_diffs)
```

```
Out[35]: (array([   20.,   118.,   556.,  1542.,  2499.,  2668.,  1776.,   645.,
                   153.,    23.]),
          array([ -4.37249748e-03,  -3.51219839e-03,  -2.65189930e-03,
                  -1.79160021e-03,  -9.31301114e-04,  -7.10020223e-05,
                   7.89297069e-04,   1.64959616e-03,   2.50989525e-03,
                   3.37019434e-03,   4.23049344e-03]),
          <a list of 10 Patch objects>)
```



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

```
In [33]: # Difference between original data set and ab_dat.csv

         act_diff = df[df['group'] == 'treatment']['converted'].mean() -  df[df['group'] == 'con
         act_diff
```

```
Out[33]: -0.0014795997940775518
```

```
In [34]: # proportion of p_diffs great than the actual difference observed

         (act_diff < p_diffs).mean()
```

```
Out[34]: 0.89829999999999999
```

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?

We are calculating the p_value. The P Value is the prrobability of getting the observed results assuming the nukll hypothesis is true.

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 [36]: import statsmodels.api as sm

         convert_old = df2.query('landing_page == "old_page" and converted == 1').shape[0]
         convert_new = df2.query('landing_page == "new_page" and converted == 1').shape[0]
         n_old = df2.query('landing_page == "old_page"').shape[0]
         n_new = df2.query('landing_page == "old_page"').shape[0]
```

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 [37]: z_score, p_value = sm.stats.proportions_ztest([convert_new, convert_old], [n_new, n_old

         z_score, p_value
```

```
Out[37]: (-1.2863207858559254, 0.90083443441123368)
```

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 the Z-score (1.29) falls within the range below 1.96, we cannot reject the null hypothesis.

Additionally, since the P_value is .90 (which is approxmiate to the p_value we calculated manually) is larger than the alpha value, we also cannot reject the null hypothesis.

Therefore, we can safely conclude that these built-in methods agree with our manual findings in part J and K.

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?

df2.head()

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 colun 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 [38]: df2.head()
```

Out[38]:

	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

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 [39]: df2['intercept'] = 1

         df2[['drop', 'ab_page']] = pd.get_dummies(df2['group'])
         df2.drop(['drop'], axis=1, inplace=True)
         df2.head()
```

Out[39]:

	user_id	timestamp	group	landing_page	converted	\
0	851104	2017-01-21 22:11:48.556739	control	old_page	0	
1	804228	2017-01-12 08:01:45.159739	control	old_page	0	
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0	
3	853541	2017-01-08 18:28:03.143765	treatment	new_page	0	
4	864975	2017-01-21 01:52:26.210827	control	old_page	1	

```
   intercept  ab_page
0          1        0
1          1        0
2          1        1
3          1        1
4          1        0
```

```
In [40]: import statsmodels.api as sm
         logit_mod = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])
```

d. Provide the summary of your model below, and use it as necessary to answer the following questions.

```
In [41]: results = logit_mod.fit()
         results.summary()

Optimization terminated successfully.
         Current function value: 0.366118
         Iterations 6


Out[41]: <class 'statsmodels.iolib.summary.Summary'>
         """
                                 Logit Regression Results
         ==============================================================================
         Dep. Variable:              converted   No. Observations:               290585
         Model:                          Logit   Df Residuals:                   290583
         Method:                           MLE   Df Model:                            1
         Date:                Thu, 21 Jun 2018   Pseudo R-squ.:               8.085e-06
         Time:                        23:27:09   Log-Likelihood:            -1.0639e+05
         converged:                       True   LL-Null:                   -1.0639e+05
                                                 LLR p-value:                    0.1897
         ==============================================================================
                          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     -1.312      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 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**?

The p-value associated with ab_page in this regression model is **.19**, while p-values returned from the built in z-test and manual calculation were ~ **0.90**.

The null-hypothesis in a logistic regression is that there is no relationship between the dependent and independent variable, meaning there IS NO relationship between the page a user is show and the conversion rate. The alternative hypothesis would be that there IS a relationship.

The null hypothesis from part II was the likelihood of a user converting from the new page is equal to or less than conversion rate from the old page. While The alternative hypothesis was that the liklihood of a user converting from the new page was greater than from the old page.

What accounts for the large difference in p-values is that the hypothesis from part II is testing whether the new landing page recieved by the treatment group lead to more conversions than the old page. In contrast, the hypothesis from section III is testing for a correlation between variables.

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 definitely other factors that could make the model more accurate and yield better results and insights. You could test different versions in different countries, states/provinces or even zip codes. Additionally other personal factors like age, gender, marital status etc.

Possible disadvantages would be added technical complexity and misinterpretation of the data, and obsuring the more relevant insights with unnecessary details.

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 [42]: countries_df = pd.read_csv('./countries.csv')
         df_new = countries_df.set_index('user_id')\
             .join(df2.set_index('user_id'), how='inner')

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

Out[43]:

user_id	country	timestamp	group	landing_page	converted	intercept	ab_page
630000	US	2017-01-19 06:26:06.548941	treatment	new_page	0	1	1
630001	US	2017-01-16 03:16:42.560309	treatment	new_page	1	1	1
630002	US	2017-01-19 19:20:56.438330	control	old_page	0	1	0
630003	US	2017-01-12 10:09:31.510471	treatment	new_page	0	1	1
630004	US	2017-01-18 20:23:58.824994	treatment	new_page	0	1	1

```
In [44]: # Create dummy variables
         df_new[['CA', 'UK', 'US']] = pd.get_dummies(df_new['country'])

In [45]: logit_mod_new = sm.Logit(df_new['converted'], df_new[['intercept','ab_page','US', 'UK']
         results_new = logit_mod_new.fit()
         results_new.summary()

Optimization terminated successfully.
         Current function value: 0.366112
         Iterations 6


Out[45]: <class 'statsmodels.iolib.summary.Summary'>
         """
                                   Logit Regression Results
         ==============================================================================
         Dep. Variable:              converted   No. Observations:               290585
         Model:                          Logit   Df Residuals:                   290581
         Method:                           MLE   Df Model:                            3
         Date:                Thu, 21 Jun 2018   Pseudo R-squ.:               2.324e-05
         Time:                        23:27:13   Log-Likelihood:            -1.0639e+05
         converged:                       True   LL-Null:                   -1.0639e+05
                                                 LLR p-value:                    0.1758
         ==============================================================================
                          coef    std err          z      P>|z|      [0.025      0.975]
         ------------------------------------------------------------------------------
         intercept     -2.0300      0.027    -76.249      0.000      -2.082      -1.978
         ab_page       -0.0150      0.011     -1.308      0.191      -0.037       0.007
         US             0.0408      0.027      1.516      0.130      -0.012       0.093
         UK             0.0506      0.028      1.784      0.074      -0.005       0.106
         ==============================================================================
         """

In [46]: np.exp(0.0408), np.exp(0.0506)

Out[46]: (1.0416437559600236, 1.0519020483004984)
```

Since Canada was not included in the original regression it will serve as our baseline. In this case we would that US users are 4% more likely to convert and UK users are 5% more likely to convert than Canadian Users.

Given the high p-values, the effects are not statistically 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 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 [48]: df_new['new_CA'] = df_new['ab_page']*df_new['CA']
         df_new['new_UK'] = df_new['ab_page']*df_new['UK']
         df_new['new_US'] = df_new['ab_page']*df_new['US']
```

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

Out[49]:

user_id	country	timestamp	group	landing_page	converted	intercept	ab_page	CA	UK	US	new_CA	new_UK	new_US
630000	US	2017-01-19 06:26:06.548941	treatment	new_page	0	1	1	0	0	1	0	0	1
630001	US	2017-01-16 03:16:42.560309	treatment	new_page	1	1	1	0	0	1	0	0	1
630002	US	2017-01-19 19:20:56.438330	control	old_page	0	1	0	0	0	1	0	0	0
630003	US	2017-01-12 10:09:31.510471	treatment	new_page	0	1	1	0	0	1	0	0	1
630004	US	2017-01-18 20:23:58.824994	treatment	new_page	0	1	1	0	0	1	0	0	1

In [54]:
```
# Fit Linear Model + Obtain Results
lin_mod = sm.OLS(df_new['converted'], df_new[['intercept', 'ab_page', 'US', 'new_US', '
result = lin_mod.fit()
result.summary()
```

Out[54]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                            OLS Regression Results
==============================================================================
Dep. Variable:              converted   R-squared:                       0.000
Model:                            OLS   Adj. R-squared:                  0.000
Method:                 Least Squares   F-statistic:                     1.467
Date:                Thu, 21 Jun 2018   Prob (F-statistic):              0.197
Time:                        23:38:43   Log-Likelihood:                -85264.
No. Observations:              290585   AIC:                         1.705e+05
Df Residuals:                  290579   BIC:                         1.706e+05
Df Model:                           5
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      0.1188      0.004     31.057      0.000       0.111       0.126
ab_page       -0.0069      0.005     -1.277      0.202      -0.017       0.004
US             0.0018      0.004      0.467      0.641      -0.006       0.010
new_US         0.0047      0.006      0.845      0.398      -0.006       0.016
UK             0.0012      0.004      0.296      0.767      -0.007       0.009
new_UK         0.0080      0.006      1.360      0.174      -0.004       0.020
==============================================================================
Omnibus:                   125550.316   Durbin-Watson:                   2.000
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           414291.118
```

```
Skew:                          2.345   Prob(JB):                         0.00
Kurtosis:                      6.497   Cond. No.                         26.1
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specif
"""
```

```
In [57]: log_mod2 = sm.Logit(df_new['converted'], df_new[['intercept','ab_page','US','new_US', '
         results_log2 = log_mod2.fit()
         results_log2.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.366108
         Iterations 6
```

```
Out[57]: <class 'statsmodels.iolib.summary.Summary'>
         """
                                 Logit Regression Results
         ==============================================================================
         Dep. Variable:              converted   No. Observations:               290585
         Model:                          Logit   Df Residuals:                   290579
         Method:                           MLE   Df Model:                            5
         Date:                Thu, 21 Jun 2018   Pseudo R-squ.:               3.483e-05
         Time:                        23:41:42   Log-Likelihood:            -1.0639e+05
         converged:                       True   LL-Null:                   -1.0639e+05
                                                 LLR p-value:                    0.1918
         ==============================================================================
                          coef    std err          z      P>|z|      [0.025      0.975]
         ------------------------------------------------------------------------------
         intercept     -2.0040      0.036    -55.008      0.000      -2.075      -1.933
         ab_page       -0.0674      0.052     -1.297      0.195      -0.169       0.034
         US             0.0175      0.038      0.465      0.642      -0.056       0.091
         new_US         0.0469      0.054      0.872      0.383      -0.059       0.152
         UK             0.0118      0.040      0.296      0.767      -0.066       0.090
         new_UK         0.0783      0.057      1.378      0.168      -0.033       0.190
         ==============================================================================
         """
```

The above present both linear and logist regressions for the interaction terms. Neither effect is statistically significant, given the high P-values. Furthermore, the linear plot shows an R-Squared value of zero, which suggests a bad fit.

1 Conclusion

Though a cursory skim of the data shows a difference in conversion rates between the new and old pages, there is just not enough evidence to reject the null hypothesis. In fact, according to the histogram show above, the new page actually does worse than the old one.

We were also able to conclude that the poor conversion rate was not dependant on countries, the rate between US and UK being similar. Testing conditions were also excellent, with each user having roughly a 50% chance to recieve the new or old page. There were enough instances and the sample size large enough to confidently reccommend that more data collection would not be an efficient use of resources.

My reccommendation would be to invest in a new redesign before attempting a new test and perhaps some additional survey work to discover why the page performed poorly.

Finishing Up

> Congratulations! You have reached the end of the A/B Test Results project! This is the final project in Term 1. 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.

1.1 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 [37]: from subprocess import call
         call(['python', '-m', 'nbconvert', 'Analyze_ab_test_results_notebook.ipynb'])
```

```
Out[37]: 0
```