Analyze_ab_test_results_notebook

July 6, 2020

0.1 Analyze A/B Test Results

You may either submit your notebook through the workspace here, or you may work from your local machine and submit through the next page. Either way assure that your code passes the project RUBRIC. **Please save regularly.**

This project will assure you have mastered the subjects covered in the statistics lessons. The hope is to have this project be as comprehensive of these topics as possible. Good luck!

0.2 Table of Contents

Introduction

A/B tests are very commonly performed by data analysts and data scientists. It is important that you get some practice working with the difficulties of these

For this project, you will be working to understand the results of an A/B test run by an e-commerce website. Your goal is to work through this notebook to help the company understand if they should implement the new page, keep the old page, or perhaps run the experiment longer to make their decision.

As you work through this notebook, follow along in the classroom and answer the corresponding quiz questions associated with each question. The labels for each classroom concept are provided for each question. This will assure you are on the right track as you work through the project, and you can feel more confident in your final submission meeting the criteria. As a final check, assure you meet all the criteria on the RUBRIC.

Part I - Probability

To get started, let's import our libraries.

```
In [164]: 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 [165]: df = pd.read_csv('ab_data.csv')
          df.head()
```

Out[165]:

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

```
In [166]: df.shape[0]
```

```
Out[166]: 294478
```

c. The number of unique users in the dataset.

```
In [167]: df['user_id'].nunique()
```

```
Out[167]: 290584
```

d. The proportion of users converted.

```
In [168]: df[df['converted'] == 1].shape[0]/df.shape[0]
```

```
Out[168]: 0.11965919355605512
```

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

```
In [169]: df[(df['group'] == 'treatment') & (df['landing_page'] == 'old_page')].shape[0]
```

```
Out[169]: 1965
```

```
In [170]: df[(df['group'] == 'control') & (df['landing_page'] == 'new_page')].shape[0]
```

```
Out[170]: 1928
```

f. Do any of the rows have missing values?

```
In [171]: 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
```

2. For the rows where **treatment** does not match with **new_page** or **control** does not match with **old_page**, we cannot be sure if this row truly received the new or old page. Use **Quiz 2** in the classroom to figure out how we should handle these rows.

a. Now use the answer to the quiz to create a new dataset that meets the specifications from the quiz. Store your new dataframe in **df2**.

```
In [172]: df.head()
```

Out[172]:

	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 [173]: df2 = df[(df['group'] == 'control') & (df['landing_page'] == 'old_page') | (df['group'
          df2.head()
```

Out[173]:

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

```
Out[174]: 0
```

3. Use **df2** and the cells below to answer questions for **Quiz3** in the classroom.

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

```
In [175]: df2['user_id'].nunique()
```

```
Out[175]: 290584
```

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

```
In [176]: df2.user_id[1899]
```

```
Out[176]: 773192
```

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

```
In [177]: df2[df2['user_id']==773192].index.values.astype(int)
```

```
Out[177]: array([1899, 2893])
```

```
In [178]: df2[df2['user_id'] == 773192]
```

```
Out[178]:       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
```

```
In [179]: df2[df2['user_id'] == 773192].index
```

```
Out[179]: Int64Index([1899, 2893], dtype='int64')
```

```
In [180]: df2.shape[0]
```

```
Out[180]: 290585
```

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

```
In [181]: df2 = df2.drop_duplicates(subset='user_id', keep='first')
          df2.shape[0]
```

```
Out[181]: 290584
```

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

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

```
In [182]: df2.head(5)
```

```
Out[182]:    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 [183]: df2.converted.mean()
```

```
Out[183]: 0.11959708724499628
```

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

```
In [184]: df2.query("group == 'control'").converted.mean()

Out[184]: 0.1203863045004612
```

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

```
In [185]: df2.query("group == 'treatment'").converted.mean()

Out[185]: 0.11880806551510564
```

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

```
In [188]: len(df2.query("landing_page == 'new_page'"))/len(df2)

Out[188]: 0.5000619442226688
```

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

Your answer goes here.
By analyzing the results:

- The Probability of converting regardless of page is: 11.96%
- Given an individual received the control page, the probability of converting is: 12.04% (17723 people in the control group converted)
- Given that an individual received the treatment page, the probability of converting is: 11.88% (17514 people in the treatment group converted)
- The probability of receiving the new page is: 50.01%
- The first positive thing to mention is, that the users received the new or the old page in a ration very close to 50/50. The probabilities of converting in the control group and the treatment group are very close to each other, with a difference of 0.16%.

This small difference could also appear by chance, therefore we don't have sufficient evidence to conclude that the new treatment page leads to more conversions than the old page.

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.

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

2. Assume under the null hypothesis, p_{new} and p_{old} both have "true" success rates equal to the **converted** success rate regardless of page - that is p_{new} and p_{old} are equal. Furthermore, assume they are equal to the **converted** rate in **ab_data.csv** regardless of the page.

Use a sample size for each page equal to the ones in **ab_data.csv**.

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

Use the cells below to provide the necessary parts of this simulation. If this doesn't make complete sense right now, don't worry - you are going to work through the problems below to complete this problem. You can use **Quiz 5** in the classroom to make sure you are on the right track.

a. What is the **conversion rate** for p_{new} under the null?

```
In [189]: p_new = df2.converted.mean()
          p_new

Out[189]: 0.11959708724499628
```

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

```
In [190]: p_old = df2.converted.mean()
          p_old

Out[190]: 0.11959708724499628
```

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

```
In [193]: n_new = df2.query("group == 'treatment'").user_id.nunique()
          n_new

Out[193]: 145310
```

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

```
In [194]: n_old = df2.query("group == 'control'").user_id.nunique()
          n_old

Out[194]: 145274
```

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

```
In [195]: #simulate n transactions with a conversion rate of p with np.random.choice
          new_page_converted = np.random.choice([1,0], size = n_new, replace = True, p = (p_new,
          new_page_converted.sum()
```

```
Out[195]: 17346
```

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

```
In [196]: old_page_converted = np.random.choice([1,0], size = n_old, replace = True, p = [p_old,
          old_page_converted.sum()
```

```
Out[196]: 17267
```

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

```
In [197]: new_page_converted.mean() - old_page_converted.mean()
```

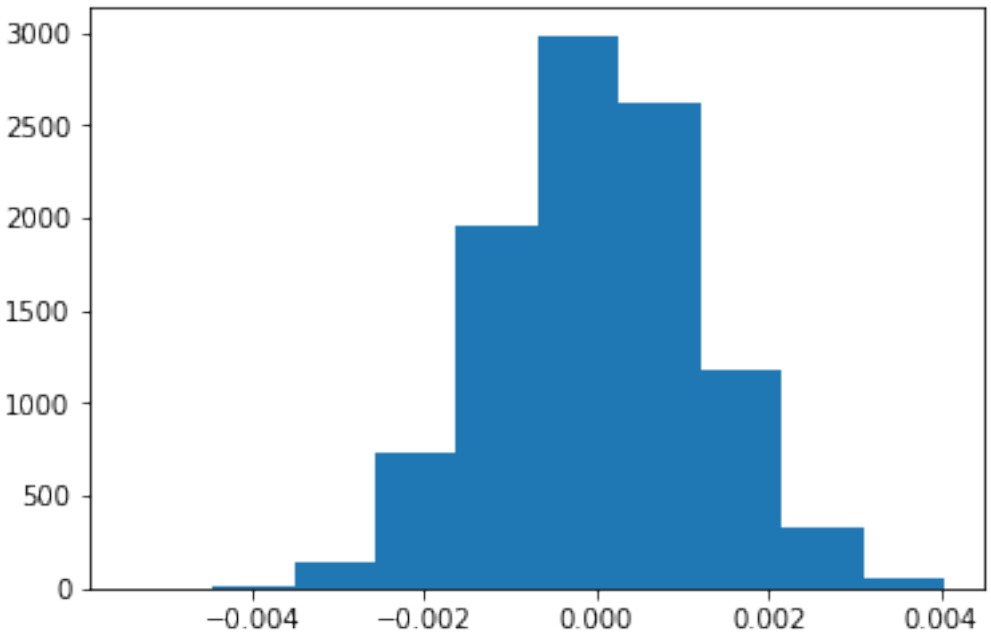
```
Out[197]: 0.0005142186107165575
```

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

```
In [198]: #creating the sampling distribution with 10000 simulations of the steps before
          p_diffs = []
          for _ in range(10000):
              new_page_converted = np.random.choice([1,0], size = n_new, replace = True, p = (p_
              old_page_converted = np.random.choice([1,0], size = n_old, replace = True, p = (p_
              diff = new_page_converted.mean() - old_page_converted.mean()
              p_diffs.append(diff)
```

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 [199]: plt.hist(p_diffs);
```



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

```
In [201]: num_conv_treat = df2.query("group == 'treatment' and converted == 1").count()[0]
          num_conv_treat
```

```
Out[201]: 17264
```

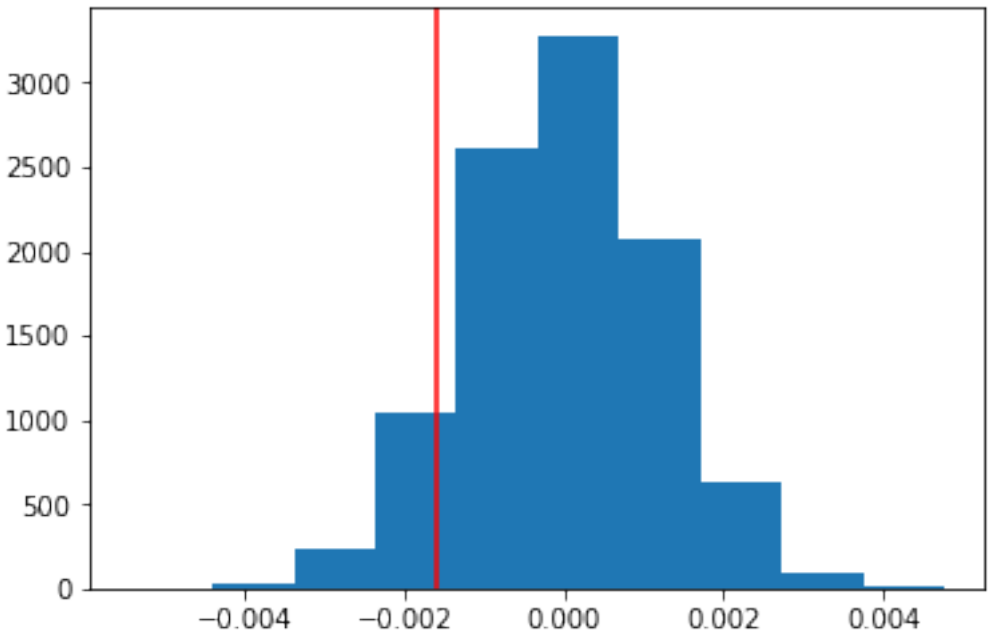
```
In [202]: num_conv_control = df2.query("group == 'control' and converted == 1").count()[0]
          num_conv_control
```

```
Out[202]: 17489
```

```
In [204]: p_actual_old = df2.query("group == 'control'").converted.mean()
          p_actual_new = df2.query("group == 'treatment'").converted.mean()
          actual_diff = p_actual_new - p_actual_old
          print("Number of converted persons in control group: ",num_conv_control, "| p_old: ",
          print("Number of converted persons in treatment group: ",num_conv_treat, "| p_new: ",
          print("Actual difference: ", actual_diff);
```

```
Number of converted persons in control group:  17489 | p_old:  0.1203863045
Number of converted persons in treatment group:  17264 | p_new:  0.118808065515
Actual difference:  -0.00157823898536
```

```
In [206]: p_diffs = np.array(p_diffs)
          #calcualte the null_vals based on the std of the p_diffs array
          null_vals = np.random.normal(0, p_diffs.std(), p_diffs.size)
          plt.hist(null_vals);
          plt.axvline(actual_diff, color = 'r');
```



```
In [207]: (null_vals > actual_diff).mean()
```

```
Out[207]: 0.90539999999999998
```

k. Please explain using the vocabulary you've learned in this course what you just computed in part

Answer We assumed that the null hypothesis is true.

With that, we assume that

$$p_{new} = p_{old}$$

so both pages have the same converting rates over the whole sample.

Therefore we also assume, that the individual converting probability of each page is equal to the one of the whole sample.

Based on that, we bootstrapped a sampling distribution for both pages and calculated the differences in the converting probability per page with n equal to the original number of people who received each page and a converting probability of 0.119597.

With the resulting standard deviation of the differences (which is coming from the simulated population), we then calcualted values coming from a normal distribution around 0.

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?

Answer:

This Value is called **p-value**.

As last step we calculated the proportion of values which are bigger than the actually observed difference.

- The calculated p-value now tells us the probability of receiving this observed statistic if the null hypothesis is true.
- With a Type-I-Error-Rate of 0.05, we can say that 0.9095 > 0.05, therefore we don't have enough evidence to reject the null hypothesis.

l. We could also use a built-in to achieve similar results. Though using the built-in might be easier to code, the above portions are a walkthrough of the ideas that are critical to correctly thinking about statistical significance. Fill in the below to calculate the number of conversions for each page, as well as the number of individuals who received each page. Let `n_old` and `n_new` refer the the number of rows associated with the old page and new pages, respectively.

```
In [208]: 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("landing_page == 'old_page'").count()[0]
          n_new = df2.query("landing_page == 'new_page'").count()[0]
```

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

m. Now use `stats.proportions_ztest` to compute your test statistic and p-value. Here is a helpful link on using the built in.

```
In [209]: from scipy.stats import norm

          #calculate z-test
          z_score, p_value = sm.stats.proportions_ztest([convert_old, convert_new], [n_old, n_ne

          #calculate the critical z_term
          z_critical=norm.ppf(1-(0.05))

          print("Z-Score: ",z_score, "\nCritical Z-Score: ", z_critical, "\nP-Value: ", p_value)
```

```
Z-Score:  1.31092419842
Critical Z-Score:  1.64485362695
P-Value:  0.905058312759
```

n. What do the z-score and p-value you computed in the previous question mean for the conversion rates of the old and new pages? Do they agree with the findings in parts **j.** and **k.**?

Answer: The p-value here agrees with our findings in j.

- Also the calculated Z-Score is smaller than the Critical Z - Score, so we also fail to reject the null hypothesis based on the Z-test.
- In conclusion we accept the null hypothesis that the coversion rates of the old page are equal or better than the conversion rates of the new page.

Part III - A regression approach

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

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

Since this case is binary, a **logistic regression** should be performed.

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

```
In [210]: df_log = df2.copy()
          df_log.head()
```

```
Out[210]:    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 [211]: #add intercept
          df_log["intercept"] = 1

          #get dummies and rename
          df_log = df_log.join(pd.get_dummies(df_log['group']))
          df_log.rename(columns = {"treatment": "ab_page"}, inplace=True)
```

```
In [212]: df_log.head()
```

```
Out[212]:    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  control  ab_page
          0          1        1        0
          1          1        1        0
          2          1        0        1
          3          1        0        1
          4          1        1        0
```

c. Use **statsmodels** to instantiate your regression model on the two columns you created in part b., then fit the model using the two columns you created in part **b.** to predict whether or not an individual converts.

```
In [213]: y = df_log["converted"]
          x = df_log[["intercept", "ab_page"]]

          #load model
          log_mod = sm.Logit(y,x)

          #fit model
          result = log_mod.fit()
```

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

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

```
In [216]: result.summary2()

Out[216]: <class 'statsmodels.iolib.summary2.Summary'>
          """
                                Results: Logit
          ================================================================
          Model:              Logit            No. Iterations:   6.0000
          Dependent Variable: converted        Pseudo R-squared: 0.000
          Date:               2020-07-06 22:22 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]
          ----------------------------------------------------------------
          intercept    -1.9888    0.0081  -246.6690  0.0000  -2.0046  -1.9730
          ab_page      -0.0150    0.0114    -1.3109  0.1899  -0.0374   0.0074
          ================================================================

          """
```

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

Answer:
The **p-value** associated with ab_page is 0.1899. This is because the approach of calculating the p-value is different for each case.

- For the first case we calculate the probability receiving a observed statistic if the null hypothesis is true. Therefore this is a one-sided test.
- However, the ab_page p-value is the result of a two sided test, because the null hypothesis for this case is, that there is no significant relationship between the conversion rate and ab_page.
- Therefore give us a variable with a low p value "a meaningful addition to your model because changes in the predictor's value are related to changes in the response variable"

(http://blog.minitab.com/blog/adventures-in-statistics-2/how-to-interpret-regression-analysis-results-p-values-and-coefficients).

- Based on that p_value we can say, that the conversion is not **significant** dependent on the page.

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?

Answer:

Other features to consider could be extracts of the time stamp, for example the day of the week or the gender/income infrastructure (if this data would be available). This could lead to more precise results and a higher accuracy.

The disadvantages are the increasing complexity of interpretation and the possible introduction of multicollinearity. However, the last problem can be solved with calculating the **VIF's**.

g. Now along with testing if the conversion rate changes for different pages, also add an effect based on which country a user lives in. You will need to read in the **countries.csv** dataset and merge together your datasets on the appropriate rows. Here are the docs for joining tables.

Does it appear that country had an impact on conversion? Don't forget to create dummy variables for these country columns - **Hint: You will need two columns for the three dummy variables.** Provide the statistical output as well as a written response to answer this question.

```
In [217]: df_countries = pd.read_csv("countries.csv")
          df_countries.head()
```

Out[217]:

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

```
In [218]: #merge the dataframes together
          df_log_country = df_log.merge(df_countries, on="user_id", how = "left")

In [219]: df_log_country = df_log_country.join(pd.get_dummies(df_log_country['country']))

In [220]: df_log_country.head()
```

Out[220]:

	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	control	ab_page	country	CA	UK	US
0	1	1	0	US	0	0	1
1	1	1	0	US	0	0	1
2	1	0	1	US	0	0	1
3	1	0	1	US	0	0	1
4	1	1	0	US	0	0	1

```
In [222]: y = df_log_country["converted"]
          X = df_log_country[["intercept", "ab_page", "CA", "UK"]]
```

```
log_mod = sm.Logit(y,X)
results = log_mod.fit()
results.summary2()
```

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

Out[222]:
```
<class 'statsmodels.iolib.summary2.Summary'>
"""
                          Results: Logit
=================================================================
Model:              Logit            No. Iterations:   6.0000
Dependent Variable: converted        Pseudo R-squared: 0.000
Date:               2020-07-06 22:27 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]
-----------------------------------------------------------------
intercept    -1.9893    0.0089  -223.7628  0.0000  -2.0067  -1.9718
ab_page      -0.0149    0.0114    -1.3069  0.1912  -0.0374   0.0075
CA           -0.0408    0.0269    -1.5161  0.1295  -0.0934   0.0119
UK            0.0099    0.0133     0.7433  0.4573  -0.0162   0.0359
=================================================================

"""
```

ab_page reciprocal exponential:

In [223]:
```
1/np.exp(-0.0149)
```

Out[223]:
```
1.0150115583846535
```

A conversion is **1.015** times less likely, if a user receives the treatment page, holding all other variables constant

In [224]:
```
#reciprocal exponential
1/np.exp(-0.0408)
```

Out[224]:
```
1.0416437559600236
```

A conversion is **1.042** times less likely, if the user lives in CA and not the US.

In [225]:
```
#UK exponential:
np.exp(0.0099)
```

```
Out[225]: 1.0099491671175422
```

A conversion is **1.00995** times more likely, if the user lives in UK and not the US.

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 [226]: #create the interaction higher order term for the ab_page and country columns
          df_log_country["CA_page"], df_log_country["UK_page"] = df_log_country["CA"] * df_log_c

In [227]: df_log_country.head(3)
```

Out[227]:

	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	

	intercept	control	ab_page	country	CA	UK	US	CA_page	UK_page
0	1	1	0	US	0	0	1	0	0
1	1	1	0	US	0	0	1	0	0
2	1	0	1	US	0	0	1	0	0

```
In [229]: y = df_log_country["converted"]
          x = df_log_country[["intercept", "ab_page", "CA", "UK", "CA_page", "UK_page"]]

          log_mod = sm.Logit(y,x)
          results = log_mod.fit()
          results.summary2()

Optimization terminated successfully.
         Current function value: 0.366109
         Iterations 6


Out[229]: <class 'statsmodels.iolib.summary2.Summary'>
          """
                                  Results: Logit
          =================================================================
          Model:              Logit            No. Iterations:   6.0000
          Dependent Variable: converted        Pseudo R-squared: 0.000
          Date:               2020-07-06 22:34 AIC:              212782.6602
          No. Observations:   290584           BIC:              212846.1381
          Df Model:           5                Log-Likelihood:   -1.0639e+05
          Df Residuals:       290578           LL-Null:          -1.0639e+05
          Converged:          1.0000           Scale:            1.0000
          -----------------------------------------------------------------
                        Coef.   Std.Err.      z      P>|z|    [0.025   0.975]
          -----------------------------------------------------------------
```

```
intercept    -1.9865    0.0096  -206.3440  0.0000  -2.0053  -1.9676
ab_page      -0.0206    0.0137    -1.5052  0.1323  -0.0473   0.0062
CA           -0.0175    0.0377    -0.4652  0.6418  -0.0914   0.0563
UK           -0.0057    0.0188    -0.3057  0.7598  -0.0426   0.0311
CA_page      -0.0469    0.0538    -0.8718  0.3833  -0.1523   0.0585
UK_page       0.0314    0.0266     1.1807  0.2377  -0.0207   0.0835
==================================================================

"""
```

Conclusion : Based on these results, we can see that the p_values for the interaction terms are definietly not significant and even decrease the significance of the original "CA" and "UK" columns. Therefore we should not include these higher order terms in our model.

Finishing Up

> Congratulations! You have reached the end of the A/B Test Results project! You should be very proud of all you have accomplished!
>
> **Tip**: Once you are satisfied with your work here, check over your report to make sure that it is satisfies all the areas of the rubric (found on the project submission page at the end of the lesson). You should also probably remove all of the "Tips" like this one so that the presentation is as polished as possible.

0.3 Directions to Submit

> Before you submit your project, you need to create a .html or .pdf version of this notebook in the workspace here. To do that, run the code cell below. If it worked correctly, you should get a return code of 0, and you should see the generated .html file in the workspace directory (click on the orange Jupyter icon in the upper left).
>
> Alternatively, you can download this report as .html via the **File** > **Download as** submenu, and then manually upload it into the workspace directory by clicking on the orange Jupyter icon in the upper left, then using the Upload button.
>
> Once you've done this, you can submit your project by clicking on the "Submit Project" button in the lower right here. This will create and submit a zip file with this .ipynb doc and the .html or .pdf version you created. Congratulations!

```
In [230]: from subprocess import call
          call(['python', '-m', 'nbconvert', 'Analyze_ab_test_results_notebook.ipynb'])
```

```
Out[230]: 0
```

```
In [ ]:
```