Analyze_ab_test_results_notebook

March 25, 2019

0.1 Analyze A/B Test Results

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

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

0.2 Table of Contents

### Introduction

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

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

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

#### Part I - Probability

To get started, let's import our libraries.

```
In [1]: import pandas as pd
        import numpy as np
        import random
        import matplotlib.pyplot as plt
        %matplotlib inline
        #We are setting the seed to assure you get the same answers on quizzes as we set up
        random.seed(42)
```

1. Now, read in the `ab_data.csv` data. Store it in `df`. **Use your dataframe to answer the questions in Quiz 1 of the classroom.**

a. Read in the dataset and take a look at the top few rows here:

```
In [2]: df=pd.read_csv('ab_data.csv')
        df.head()
```

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

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

```
In [3]: df.shape
```

```
Out[3]: (294478, 5)
```

c. The number of unique users in the dataset.

```
In [6]: unique_ids=pd.value_counts(df.user_id)

        unique_ids
```

```
Out[6]: 637561    2
        821876    2
        643869    2
        938802    2
        916765    2
        690255    2
        737500    2
        680018    2
        853835    2
        736746    2
        722827    2
        904340    2
        757485    2
        863300    2
        905507    2
        902109    2
        782432    2
        644294    2
        899374    2
        881704    2
        656951    2
        869729    2
```

```
720460    2
889529    2
812376    2
846972    2
776770    2
859842    2
844475    2
848746    2
         ..
874753    1
868610    1
870659    1
880900    1
876806    1
669933    1
878855    1
856328    1
858377    1
852234    1
854283    1
864524    1
696574    1
702717    1
700668    1
690427    1
688378    1
694521    1
692472    1
714999    1
712950    1
719093    1
717044    1
706803    1
704754    1
710897    1
708848    1
665839    1
663790    1
630836    1
Name: user_id, Length: 290584, dtype: int64
```

d. The proportion of users converted.

```
In [7]: pd.value_counts(df.converted)
```

```
Out[7]: 0    259241
        1     35237
        Name: converted, dtype: int64
```

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

```
In [8]: Frequency = df.groupby(['group', 'landing_page']).size().reset_index(name='Frequency')

        Frequency

Out[8]:        group landing_page  Frequency
        0    control     new_page       1928
        1    control     old_page     145274
        2  treatment     new_page     145311
        3  treatment     old_page       1965
```

f. Do any of the rows have missing values?

```
In [9]: 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 [11]: df2=df

In [12]: df2=df[((df['group'] == 'treatment') == (df['landing_page'] == 'new_page')) == True]

In [13]: df2.groupby(['group', 'landing_page']).size()

Out[13]: group      landing_page
         control    old_page        145274
         treatment  new_page        145311
         dtype: int64

In [14]: # 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[14]: 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 [15]: unique_ids2=pd.value_counts(df2.user_id)

         unique_ids2

Out[15]: 773192    2
         630732    1
         811737    1
         797392    1
         795345    1
         801490    1
         799443    1
         787157    1
         793302    1
         817882    1
         842446    1
         815835    1
         805596    1
         803549    1
         809694    1
         807647    1
         895712    1
         840399    1
         836301    1
         899810    1
         834242    1
         936604    1
         934557    1
         940702    1
         938655    1
         830144    1
         828097    1
         832195    1
         838348    1
         821956    1
                  ..
         734668    1
         736717    1
         730574    1
         775632    1
         771538    1
         642451    1
         773587    1
         783828    1
         785877    1
```

```
779734    1
781783    1
759256    1
726472    1
748999    1
746950    1
753093    1
751044    1
740803    1
738754    1
744897    1
742848    1
634271    1
632222    1
636316    1
630169    1
650647    1
648598    1
654741    1
652692    1
630836    1
Name: user_id, Length: 290584, dtype: int64
```

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

```
In [16]: 773192

Out[16]: 773192
```

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

```
In [28]: df2.loc[df['user_id'] == 773192]

Out[28]:       user_id                   timestamp      group landing_page  converted
         1899   773192  2017-01-09 05:37:58.781806  treatment     new_page          0
         2893   773192  2017-01-14 02:55:59.590927  treatment     new_page          0
```

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

```
In [35]: df2.drop(df.index[2893], inplace=True)


        ---------------------------------------------------------------------------

        KeyError                                  Traceback (most recent call last)

        <ipython-input-35-53e7047f2f10> in <module>()
    ----> 1 df2.drop(df.index[2893], inplace=True)
    /opt/conda/lib/python3.6/site-packages/pandas/core/frame.py in drop(self, labels, axis,
   3695                                            index=index, columns=columns,
   3696                                            level=level, inplace=inplace,
-> 3697                                            errors=errors)
   3698
   3699     @rewrite_axis_style_signature('mapper', [('copy', True),


    /opt/conda/lib/python3.6/site-packages/pandas/core/generic.py in drop(self, labels, axis
   3109         for axis, labels in axes.items():
   3110             if labels is not None:
-> 3111                 obj = obj._drop_axis(labels, axis, level=level, errors=errors)
   3112
   3113         if inplace:


    /opt/conda/lib/python3.6/site-packages/pandas/core/generic.py in _drop_axis(self, labels
   3141                 new_axis = axis.drop(labels, level=level, errors=errors)
   3142             else:
-> 3143                 new_axis = axis.drop(labels, errors=errors)
   3144             result = self.reindex(**{axis_name: new_axis})
   3145


    /opt/conda/lib/python3.6/site-packages/pandas/core/indexes/base.py in drop(self, labels,
   4402             if errors != 'ignore':
   4403                 raise KeyError(
-> 4404                     '{} not found in axis'.format(labels[mask]))
   4405             indexer = indexer[~mask]
   4406         return self.delete(indexer)


    KeyError: '[2893] not found in axis'
```

```
In [36]: df2.loc[df2['user_id'] == 773192]

Out[36]:       user_id                   timestamp      group landing_page  converted
         1899   773192  2017-01-09 05:37:58.781806  treatment     new_page          0
```

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 [47]: df2.head(20)

Out[47]:     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
5    936923  2017-01-10 15:20:49.083499    control     old_page          0
6    679687  2017-01-19 03:26:46.940749  treatment     new_page          1
7    719014  2017-01-17 01:48:29.539573    control     old_page          0
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
10   929503  2017-01-18 05:37:11.527370  treatment     new_page          0
11   834487  2017-01-21 22:37:47.774891  treatment     new_page          0
12   803683  2017-01-09 06:05:16.222706  treatment     new_page          0
13   944475  2017-01-22 01:31:09.573836  treatment     new_page          0
14   718956  2017-01-22 11:45:11.327945  treatment     new_page          0
15   644214  2017-01-22 02:05:21.719434    control     old_page          1
16   847721  2017-01-17 14:01:00.090575    control     old_page          0
17   888545  2017-01-08 06:37:26.332945  treatment     new_page          1
18   650559  2017-01-24 11:55:51.084801    control     old_page          0
19   935734  2017-01-17 20:33:37.428378    control     old_page          0
```

```
In [37]: df2.converted.mean()

Out[37]: 0.11959791040050657
```

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

```
In [38]: df2[df2['group']=='control']['converted'].mean()

Out[38]: 0.1203863045004612

In [135]: ((df2.group == 'control') & (df2.converted == 1)).mean()

Out[135]: 0.060186109256595385
```

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

```
In [39]: df2[df2['group']=='treatment']['converted'].mean()

Out[39]: 0.11880970077352933

In [136]: ((df2.group == 'treatment') & (df2.converted == 1)).mean()

Out[136]: 0.059411801143911182
```

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

```
In [62]: (df2[df2['landing_page']=='new_page'].count())/(df2.landing_page.count())
```

```
Out[62]: user_id         0.500059
         timestamp       0.500059
         group           0.500059
         landing_page    0.500059
         converted       0.500059
         dtype: float64
```

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.

No. The results are too similar to suggest one choice over the other.
### 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.

$p_{old} : \mu \leq 0$ $p_{new} : \mu > 0$

`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 [86]: p_new = (df2.converted == 1).mean()
```

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

```
In [87]: p_old = (df2.converted == 1).mean()
```

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

```
In [158]: n_new = df2[df2['group']=='treatment']['converted'].count()+2
          n_new

Out[158]: 145310
```

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

```
In [160]: n_old = df2[df2['group']=='control']['converted'].count()
          n_old
```

```
Out[160]: 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 [161]: new_page_converted = np.random.choice([0, 1], size=n_new, p=[1-p_new, p_new])
```

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 [166]: old_page_converted = np.random.choice([0, 1], size=n_old, p=[1-p_old, p_old])
```

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

```
In [168]: new_page_converted.mean()-old_page_converted.mean()
```

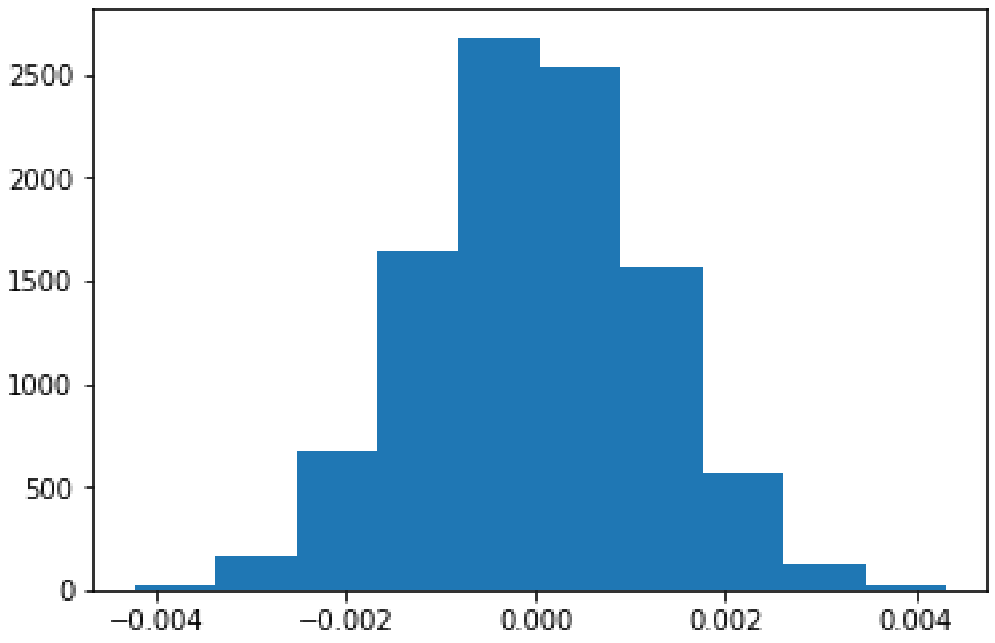
```
Out[168]: 0.00016994931067196295
```

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 [169]: p_diffs = []
          for _ in range(10000):
              new_page_converted = np.random.binomial(n_new,p_new)
              old_page_converted = np.random.binomial(n_old, p_old)
              diff = new_page_converted/n_new - old_page_converted/n_old
              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.

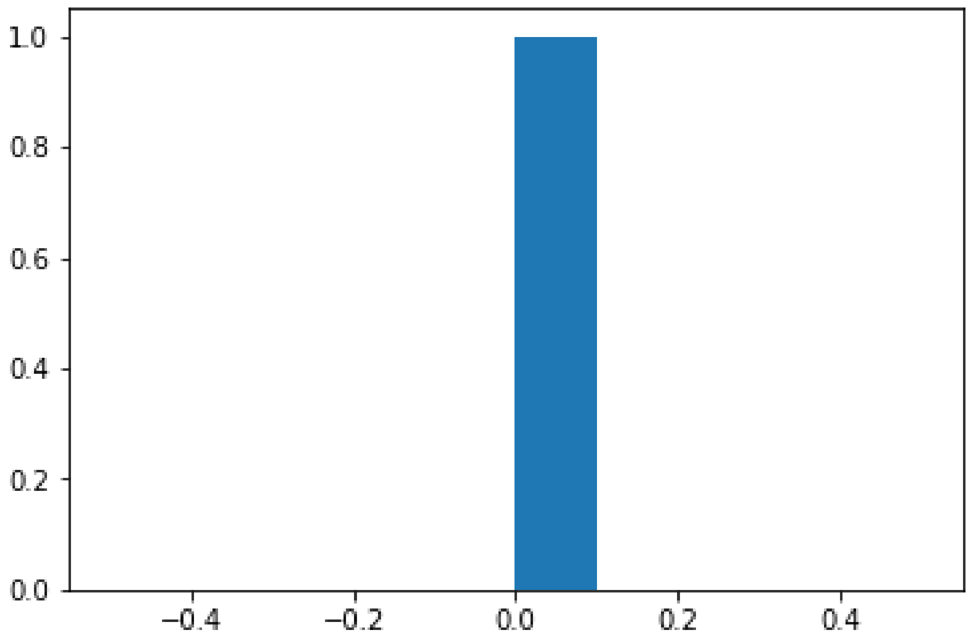
```
In [170]: plt.hist(p_diffs);
```



```
In [171]: obs_diffs=np.array(diff)

In [172]: null_vals = np.random.normal(0, obs_diffs.std(), obs_diffs.size)

In [173]: plt.hist(null_vals);
```



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

```
In [174]: (null_vals > p_diffs).mean()

Out[174]: 0.50019999999999998

In [178]: (p_diffs > obs_diffs).mean()

Out[178]: 0.93230000000000002
```

k. Please explain using the vocabulary you've learned in this course what you just computed in part **j.** What is this value called in scientific studies? What does this value mean in terms of whether or not there is a difference between the new and old pages?

In part j I calculated the probability value, or p-value, by simulating distribution under the null hypothesis to find if our observed statistics comes from the null or alternative 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 [176]: import statsmodels.api as sm

          convert_old = sum((df2.group == 'control') & (df2.converted == 1))
          convert_new = sum((df2.group == 'treatment') & (df2.converted == 1))
          n_old = sum(df2.group == 'control')
          n_new = sum(df2.group == 'treatment')
```

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 [177]: z_score, p_value = sm.stats.proportions_ztest([convert_new, convert_old], [n_new, n_ol
          z_score, p_value

Out[177]: (-1.3095575124241912, 0.90482721517044784)
```

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.**?

The p values from j and k are similar to the p value above. This evidence would suggest strongly to reject the null.

### Part III - A regression approach

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

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

Logistic Regression

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

```
In [186]: df['intercept']=1
          df[['not_ab_page','ab_page']]= pd.get_dummies(df['group'])
          df2=df.drop('not_ab_page', axis=1)
          df2.head()
```

```
Out[186]:    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

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

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

```
In [189]: df2['intercept']=1
          logit_mod=sm.Logit(df2['converted'],df2[['intercept','ab_page']])
          results=logit_mod.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.366243
         Iterations 6
```

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

```
In [190]: results.summary()
```

```
Out[190]: <class 'statsmodels.iolib.summary.Summary'>
          """
                                     Logit Regression Results
          ==============================================================================
          Dep. Variable:              converted   No. Observations:               294478
          Model:                          Logit   Df Residuals:                   294476
          Method:                           MLE   Df Model:                            1
          Date:                Tue, 19 Mar 2019   Pseudo R-squ.:               7.093e-06
          Time:                        20:47:00   Log-Likelihood:            -1.0785e+05
          converged:                       True   LL-Null:                   -1.0785e+05
                                                  LLR p-value:                    0.2161
          ==============================================================================
                           coef    std err          z      P>|z|      [0.025      0.975]
          ------------------------------------------------------------------------------
          intercept     -1.9887      0.008   -248.297      0.000      -2.004      -1.973
          ab_page       -0.0140      0.011     -1.237      0.216      -0.036       0.008
          ==============================================================================
          """
```

e. What is the p-value associated with **ab_page**? Why does it differ from the value you found in **Part II**? **Hint**: What are the null and alternative hypotheses associated with your regression model, and how do they compare to the null and alternative hypotheses in **Part II**?

The p value here is .216, dramatically lower than the p value in Part II, but still high above the .05 margin for error and thus we reject the null. I believe these numbers differ so greatly due to Part II being a single-tailed test and Part III being a two-tailed test (of the coefficient of ab_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?

It is usually important to consinder other factors in your regression models to improve accuracy and precision. We are not differentiating between old and new pages. With this regression model - Logistic - you are only weighing the possibility of two outcomes. So any additional term that would require multiple outcomes would force an alternative regression model.

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 [195]: countries = pd.read_csv('countries.csv')
```

```
In [196]: countries.head()
```

```
Out[196]:    user_id country
          0   834778      UK
          1   928468      US
          2   822059      UK
          3   711597      UK
          4   710616      UK
```

```
In [199]: pd.value_counts(countries.country)
```

```
Out[199]: US    203619
          UK     72466
          CA     14499
          Name: country, dtype: int64
```

```
In [232]: countries[['CA','UK', 'US']] = pd.get_dummies(countries['country'])
```

```
In [233]: countries.head()
```

```
Out[233]:    user_id country  CA  UK  US
          0   834778      UK   0   1   0
          1   928468      US   0   0   1
          2   822059      UK   0   1   0
          3   711597      UK   0   1   0
          4   710616      UK   0   1   0
```

```
In [234]: df2.head()
```

```
Out[234]:    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

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

```
In [235]: df3=df2.set_index('user_id').join(countries.set_index('user_id'))
```

```
In [243]: logit_mod2 = sm.Logit(df3.converted, df3[['intercept', 'UK', 'US']])
          results2=logit_mod2.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.366241
         Iterations 6
```

```
In [244]: results2.summary()

Out[244]: <class 'statsmodels.iolib.summary.Summary'>
          """
                                     Logit Regression Results
          ==============================================================================
          Dep. Variable:              converted   No. Observations:               294478
          Model:                          Logit   Df Residuals:                   294475
          Method:                           MLE   Df Model:                            2
          Date:                Tue, 19 Mar 2019   Pseudo R-squ.:               1.205e-05
          Time:                        22:01:31   Log-Likelihood:            -1.0785e+05
          converged:                       True   LL-Null:                   -1.0785e+05
                                                  LLR p-value:                    0.2726
          ==============================================================================
                           coef    std err          z      P>|z|      [0.025      0.975]
          ------------------------------------------------------------------------------
          intercept     -2.0319      0.026    -78.845      0.000      -2.082      -1.981
          UK             0.0450      0.028      1.599      0.110      -0.010       0.100
          US             0.0357      0.027      1.340      0.180      -0.017       0.088
          ==============================================================================
          """

In [237]: df3[['new_page', 'old_page']]= pd.get_dummies(df3['landing_page'])
          df3=df3.drop('old_page', axis=1)
          df3.head()

Out[237]:                           timestamp      group landing_page  converted  \
          user_id
          630000   2017-01-19 06:26:06.548941  treatment     new_page          0
          630001   2017-01-16 03:16:42.560309  treatment     new_page          1
          630002   2017-01-19 19:20:56.438330    control     old_page          0
          630003   2017-01-12 10:09:31.510471  treatment     new_page          0
          630004   2017-01-18 20:23:58.824994  treatment     new_page          0

                   ab_page  intercept country  CA  UK  US  new_page
          user_id
          630000         1          1      US   0   0   1         1
          630001         1          1      US   0   0   1         1
          630002         0          1      US   0   0   1         0
          630003         1          1      US   0   0   1         1
          630004         1          1      US   0   0   1         1

In [240]: logit_mod3 = sm.Logit(df3.converted, df3[['intercept', 'new_page']])
          results3=logit_mod3.fit()

Optimization terminated successfully.
         Current function value: 0.366242
         Iterations 6
```

```
In [242]: results3.summary()

Out[242]: <class 'statsmodels.iolib.summary.Summary'>
          """
                                     Logit Regression Results
          ==============================================================================
          Dep. Variable:              converted   No. Observations:               294478
          Model:                          Logit   Df Residuals:                   294476
          Method:                           MLE   Df Model:                            1
          Date:                Tue, 19 Mar 2019   Pseudo R-squ.:               8.680e-06
          Time:                        22:00:45   Log-Likelihood:            -1.0785e+05
          converged:                       True   LL-Null:                   -1.0785e+05
                                                  LLR p-value:                    0.1712
          ==============================================================================
                           coef    std err          z      P>|z|      [0.025      0.975]
          ------------------------------------------------------------------------------
          intercept     -1.9879      0.008   -248.305      0.000      -2.004      -1.972
          new_page      -0.0155      0.011     -1.368      0.171      -0.038       0.007
          ==============================================================================
          """
```

The summaries of both country and page on conversion suggest to reject the null

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 [251]: df3[['control', 'treatment']]= pd.get_dummies(df3['group'])
          df3=df3.drop('control', axis=1)

In [252]: df3.head()

Out[252]:                          timestamp      group landing_page  converted  \
          user_id
          630000   2017-01-19 06:26:06.548941  treatment     new_page          0
          630001   2017-01-16 03:16:42.560309  treatment     new_page          1
          630002   2017-01-19 19:20:56.438330    control     old_page          0
          630003   2017-01-12 10:09:31.510471  treatment     new_page          0
          630004   2017-01-18 20:23:58.824994  treatment     new_page          0

                   ab_page  intercept country  CA  UK  US  new_page  treatment
          user_id
          630000         1          1      US   0   0   1         1          1
          630001         1          1      US   0   0   1         1          1
          630002         0          1      US   0   0   1         0          0
          630003         1          1      US   0   0   1         1          1
          630004         1          1      US   0   0   1         1          1
```

```
In [258]: df3['country_page_us']=df3['treatment']*df3['US']

In [259]: df3['country_page_ca']=df3['treatment']*df3['CA']

In [260]: df3['country_page_uk']=df3['treatment']*df3['UK']

In [261]: df3.head()

Out[261]:                          timestamp      group landing_page  converted  \
          user_id
          630000   2017-01-19 06:26:06.548941  treatment     new_page          0
          630001   2017-01-16 03:16:42.560309  treatment     new_page          1
          630002   2017-01-19 19:20:56.438330    control     old_page          0
          630003   2017-01-12 10:09:31.510471  treatment     new_page          0
          630004   2017-01-18 20:23:58.824994  treatment     new_page          0

                   ab_page  intercept country  CA  UK  US  new_page  treatment  \
          user_id
          630000         1          1      US   0   0   1         1          1
          630001         1          1      US   0   0   1         1          1
          630002         0          1      US   0   0   1         0          0
          630003         1          1      US   0   0   1         1          1
          630004         1          1      US   0   0   1         1          1

                   country_page_us  country_page_ca  country_page_uk
          user_id
          630000                 1                0                0
          630001                 1                0                0
          630002                 0                0                0
          630003                 1                0                0
          630004                 1                0                0

In [263]: logit_mod4 = sm.Logit(df3.converted, df3[['intercept', 'country_page_uk', 'country_pag
          results4=logit_mod4.fit()

Optimization terminated successfully.
         Current function value: 0.366242
         Iterations 6


In [264]: results4.summary()

Out[264]: <class 'statsmodels.iolib.summary.Summary'>
          """
                                     Logit Regression Results
          ==============================================================================
          Dep. Variable:              converted   No. Observations:               294478
          Model:                          Logit   Df Residuals:                   294475
          Method:                           MLE   Df Model:                            2
```

```
Date:                Tue, 19 Mar 2019   Pseudo R-squ.:              8.331e-06
Time:                        22:20:10   Log-Likelihood:           -1.0785e+05
converged:                       True   LL-Null:                  -1.0785e+05
                                        LLR p-value:                   0.4072
=====================================================================================
                        coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------------
intercept            -1.9922      0.008   -254.566      0.000      -2.008      -1.977
country_page_uk       0.0090      0.018      0.503      0.615      -0.026       0.044
country_page_us      -0.0131      0.012     -1.055      0.292      -0.037       0.011
=====================================================================================
"""
```

This summary suggests that all three countries return data suggesting to reject the null. However, the p value for the UK's interaction with page groupings suggest conversion there is twice as high as in the US and 30% higher than in Canada.
## 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 [ ]: from subprocess import call
        call(['python', '-m', 'nbconvert', 'Analyze_ab_test_results_notebook.ipynb'])
```