Analyze_ab_test_results_notebook

November 21, 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 [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 [4]: df.nunique()
```

```
Out[4]: user_id         290584
        timestamp       294478
        group                2
        landing_page         2
        converted            2
        dtype: int64
```

d. The proportion of users converted.

```
In [5]: len(df.query('converted == 1')) / df['converted'].count()
```

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

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

```
In [6]: df[((df['group'] == 'treatment') != (df['landing_page'] == 'new_page'))== True]
```

Out[6]:

	user_id	timestamp	group	landing_page	converted
22	767017	2017-01-12 22:58:14.991443	control	new_page	0
240	733976	2017-01-11 15:11:16.407599	control	new_page	0
308	857184	2017-01-20 07:34:59.832626	treatment	old_page	0
327	686623	2017-01-09 14:26:40.734775	treatment	old_page	0
357	856078	2017-01-12 12:29:30.354835	treatment	old_page	0
490	808613	2017-01-10 21:44:01.292755	control	new_page	0
685	666385	2017-01-23 08:11:54.823806	treatment	old_page	0

713	748761	2017-01-10 15:47:44.445196	treatment	old_page	0
776	820951	2017-01-04 02:42:54.770627	treatment	old_page	0
846	637639	2017-01-11 23:09:52.682329	control	new_page	1
850	793580	2017-01-08 03:25:33.723712	control	new_page	1
889	839954	2017-01-06 20:58:22.280929	treatment	old_page	0
988	698120	2017-01-22 07:09:37.540970	control	new_page	0
1037	880442	2017-01-07 21:42:39.026815	treatment	old_page	0
1106	817911	2017-01-17 21:51:43.220160	treatment	old_page	0
1198	646342	2017-01-06 18:39:23.484797	control	new_page	0
1354	735021	2017-01-16 09:51:29.349493	control	new_page	0
1376	844475	2017-01-20 14:25:37.359614	treatment	old_page	0
1474	678638	2017-01-18 06:36:42.515395	control	new_page	0
1551	838336	2017-01-14 22:05:24.310302	treatment	old_page	0
1706	916207	2017-01-20 11:53:39.683012	treatment	old_page	0
1762	690127	2017-01-11 16:02:57.551297	treatment	old_page	1
1877	717682	2017-01-07 03:05:39.891873	control	new_page	0
2023	937692	2017-01-19 01:29:42.739007	control	new_page	0
2214	649781	2017-01-20 03:50:20.837704	control	new_page	0
2233	869707	2017-01-02 18:36:28.222510	treatment	old_page	0
2422	853156	2017-01-15 23:19:45.427866	treatment	old_page	0
2689	793494	2017-01-09 02:09:08.534282	treatment	old_page	0
2745	872666	2017-01-05 07:44:32.050781	control	new_page	0
2759	639817	2017-01-06 23:39:11.754971	control	new_page	0
...	...	...	...	...	...
292521	689329	2017-01-06 03:58:15.546309	treatment	old_page	0
292570	778969	2017-01-21 12:59:42.740399	control	new_page	1
292607	699462	2017-01-17 23:54:08.826755	treatment	old_page	0
292748	684361	2017-01-19 03:59:57.656614	control	new_page	0
292800	712112	2017-01-14 23:33:41.083796	treatment	old_page	0
292845	893018	2017-01-10 15:05:37.522921	control	new_page	0
292963	742202	2017-01-12 04:34:20.344485	treatment	old_page	0
292977	638460	2017-01-22 13:38:30.677806	treatment	old_page	0
293017	792268	2017-01-06 09:21:58.341063	control	new_page	0
293085	884635	2017-01-19 14:19:48.484389	control	new_page	0
293240	861420	2017-01-04 20:34:09.065070	treatment	old_page	0
293302	825937	2017-01-04 20:56:48.825875	treatment	old_page	0
293391	934444	2017-01-12 19:49:35.581289	treatment	old_page	0
293393	636565	2017-01-12 07:26:31.103374	control	new_page	0
293443	738761	2017-01-04 15:20:52.694440	treatment	old_page	0
293480	638376	2017-01-18 15:41:02.395882	control	new_page	0
293530	934040	2017-01-04 20:52:26.981566	treatment	old_page	0
293568	704024	2017-01-15 17:06:09.309987	control	new_page	0
293662	927109	2017-01-04 09:14:33.647192	control	new_page	0
293773	688144	2017-01-16 20:34:50.450528	treatment	old_page	1
293817	876037	2017-01-17 16:15:08.957152	treatment	old_page	1
293888	865405	2017-01-12 08:38:50.511434	control	new_page	0
293894	741581	2017-01-09 20:49:03.391764	control	new_page	0
293917	738357	2017-01-05 15:37:55.729133	treatment	old_page	0

```
293996   942612  2017-01-08 13:52:28.182648    control     new_page          0
294014   813406  2017-01-09 06:25:33.223301  treatment     old_page          0
294200   928506  2017-01-13 21:32:10.491309    control     new_page          0
294252   892498  2017-01-22 01:11:10.463211  treatment     old_page          0
294253   886135  2017-01-06 12:49:20.509403    control     new_page          0
294331   689637  2017-01-13 11:34:28.339532    control     new_page          0

[3893 rows x 5 columns]
```

```
In [7]: len(df.query('landing_page == "new_page" and group == "control"')) + len(df.query('landi
```

```
Out[7]: 3893
```

f. Do any of the rows have missing values?

```
In [7]: df.isnull().sum()
```

```
Out[7]: user_id         0
        timestamp       0
        group           0
        landing_page    0
        converted       0
        dtype: int64
```

2. For the rows where **treatment** does not match with **new_page** or **control** does not match with **old_page**, we cannot be sure if this row truly received the new or old page. 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 [8]: df2= df[((df['group'] == 'treatment') == (df['landing_page'] == 'new_page'))== True]
```

```
In [9]: df2.shape
```

```
Out[9]: (290585, 5)
```

```
In [10]: df2.head()
```

```
Out[10]:    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 [11]: # 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[11]: 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 [12]: sum(df2['user_id'].duplicated())
```

```
Out[12]: 1
```

```
In [13]: df2.nunique()
```

```
Out[13]: user_id         290584
         timestamp       290585
         group                2
         landing_page         2
         converted            2
         dtype: int64
```

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

```
In [14]: df2[df2.duplicated('user_id')]
```

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

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

The index number of duplicated row is 2893.

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

```
In [15]: df2= df2.drop_duplicates(subset='user_id')
```

```
In [16]: df2.shape
```

```
Out[16]: (290584, 5)
```

```
In [17]: df2.head()
```

```
Out[17]:    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
```

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

```
Out[18]: 0.11959708724499628
```

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

```
In [19]: df2.query('group == "control"').converted.mean()
```

```
Out[19]: 0.1203863045004612
```

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

```
In [20]: df2.query('group == "treatment"').converted.mean()
```

```
Out[20]: 0.11880806551510564
```

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

```
In [21]: df2.query('landing_page == "new_page"').count() / df2.landing_page.count()
```

```
Out[21]: user_id         0.500062
         timestamp       0.500062
         group           0.500062
         landing_page    0.500062
         converted       0.500062
         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.

There is not enough evidence to conclude that there is a significant diffrence between treatment and control group that lead more conversion. The proportion of control group siligtly higher than the treatment group. However, we do not know one page is better than the other for a certain amount time. Threfore, the A/B Test require to understand the significant diffrence between treatment and control groups.

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.

H0: Old page is better than new page or same as new page (pold => pnew) H1: new page is better than old page (pnew > pold)

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 [22]: df2.head()
```

Out[22]:

	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 [24]: pnew= df2['converted'].mean()

In [25]: pnew

Out[25]: 0.11959708724499628
```

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

```
In [26]: pold= df2['converted'].mean()

In [27]: pold

Out[27]: 0.11959708724499628
```

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

```
In [28]: n_new= len(df2.query('landing_page == "new_page"'))

In [29]: n_new

Out[29]: 145310
```

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

```
In [30]: n_old= len(df2.query('landing_page == "old_page"'))

In [31]: n_old

Out[31]: 145274
```

In [ ]:

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**.

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**.

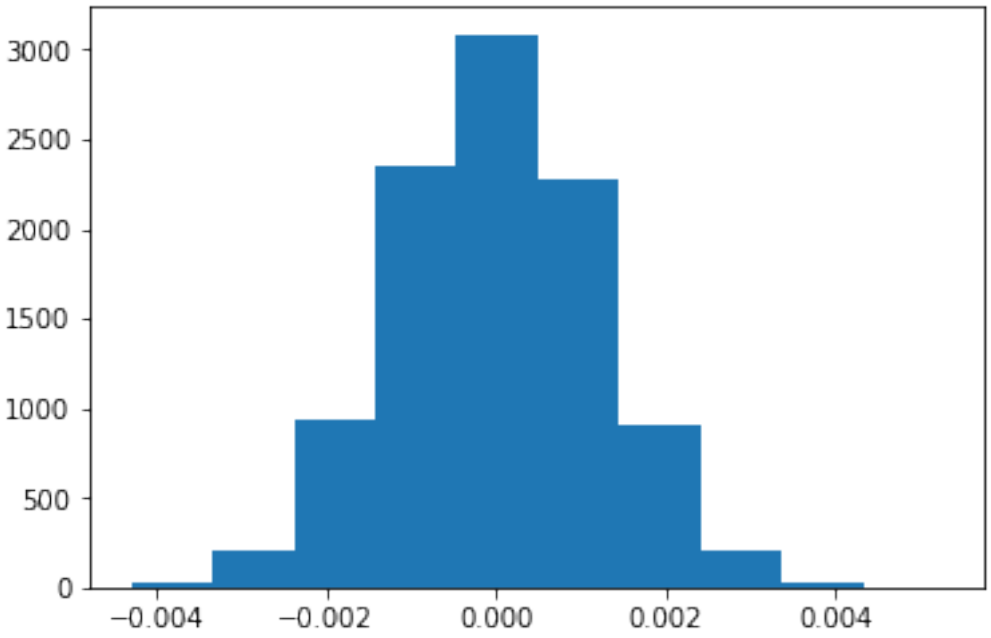
g. Find p_{new} - p_{old} for your simulated values from part (e) and (f).

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 [32]: new_page_converted = np.random.choice([0, 1], size=n_new, p=[1-pnew, pnew])

In [33]: new_page_converted

Out[33]: array([0, 0, 0, ..., 0, 0, 0])

In [34]: old_page_converted = np.random.choice([0, 1], size=n_old, p=[1-pold, pold])

In [35]: old_page_converted

Out[35]: array([0, 0, 0, ..., 0, 0, 0])

In [36]: new_page_converted.mean() - old_page_converted.mean()

Out[36]: -0.00095186545454729876

In [37]: p_diffs = []
         for _ in range(10000):
             new_page_converted = np.random.choice([0, 1], size=n_new, p=[1-pnew, pnew])
             old_page_converted = np.random.choice([0, 1], size=n_old, p=[1-pold, pold])
             p_diff = new_page_converted.mean() - old_page_converted.mean()
             p_diffs.append(p_diff)
```

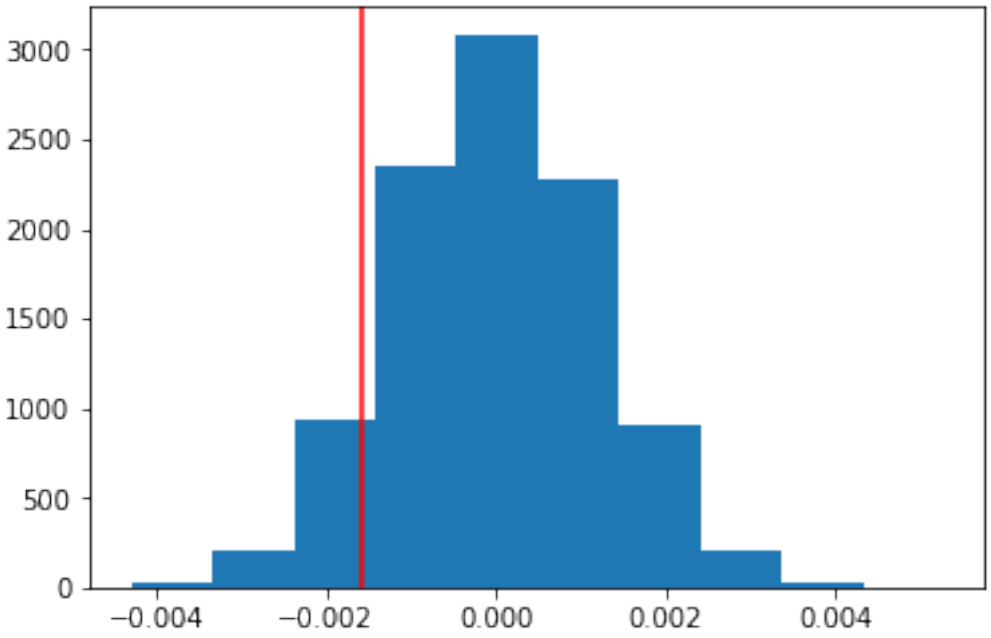
i. Plot a histogram of the **p_diffs**. Does this plot look like what you expected? Use the matching problem in the classroom to assure you fully understand what was computed here.

```
In [38]: plt.hist(p_diffs);
```



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

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

In [40]: actual_diffs

Out[40]: -0.0015782389853555567

In [41]: plt.hist(p_diffs)
         plt.axvline(actual_diffs, color='r');
```




```
In [42]: p_diffs = np.array(p_diffs)
         p_value = (p_diffs > actual_diffs).mean()
         p_value
```

```
Out[42]: 0.9083
```

```
In [ ]:
```

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?

There is no significant diffrences between the control and treatment groups' conversion rate(p> .05).

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 [43]: import statsmodels.api as sm

         convert_old = len(df2.query("landing_page == 'old_page' and converted == 1"))
         convert_new = len(df2.query("landing_page == 'new_page' and converted == 1"))
         n_old = len(df2.query("landing_page == 'old_page'"))
         n_new = len(df2.query("landing_page == 'new_page'"))
```

```
/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 [44]: import statsmodels.api as sm
         z_score, p_val = sm.stats.proportions_ztest([convert_new, convert_old], [n_new, n_old],
         print(z_score, p_val)
```

```
-1.31092419842 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.**?

There is no significant diffrence between new and old pages (z = -1.31, p> .05). Z socore (-1.31) is less tha the critical value(1.96). Therefore, we fail to reject the null hypothesis.

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

Out[45]:

	user_id	timestamp	group	landing_page	converted	intercept	ab_page
0	851104	2017-01-21 22:11:48.556739	control	old_page	0	1	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		

```
1          1        0
2          1        1
3          1        1
4          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 [46]: logit = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])
         results = logit.fit()

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

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

```
In [47]: results.summary2()

Out[47]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                 Results: Logit
         =================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2020-10-13 13:44 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**? **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 regression result indicated no significant effect between conversion and ab_page (p> .05). The p-value that is associated with ab_page is 0.189. This value is different than the value in part II because A/B test and logistic regression maintain different different hypothesizes. In part III, the

null and alternative hypotheses for the regression model are Hnull:Pnew=Pold and Halternative: PnewPold, and it is a two-tail test. In part II, A/B test, the null and alternative hypotheses for the regression model are Hnull: Pold>= Pnew and Halternative: Pnew> Pold and A/B test is a one-tail test.

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?

The other independent variables may influence the conversion rate. If we have data, we can run a regression to understand how the other independent variables affect the conversion rate.

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 [49]: df_countries= pd.read_csv("countries.csv")

In [50]: df_countries.head()

Out[50]:    user_id country
         0   834778      UK
         1   928468      US
         2   822059      UK
         3   711597      UK
         4   710616      UK

In [51]: df3 = df_countries.set_index('user_id').join(df2.set_index('user_id'))

In [52]: df3.head()

Out[52]:         country                   timestamp      group landing_page  \
         user_id
         834778       UK  2017-01-14 23:08:43.304998    control     old_page
         928468       US  2017-01-23 14:44:16.387854  treatment     new_page
         822059       UK  2017-01-16 14:04:14.719771  treatment     new_page
         711597       UK  2017-01-22 03:14:24.763511    control     old_page
         710616       UK  2017-01-16 13:14:44.000513  treatment     new_page

                  converted  intercept  ab_page
         user_id
         834778           0          1        0
         928468           0          1        1
         822059           1          1        1
         711597           0          1        0
         710616           0          1        1
```

```
In [53]: df3['country'].unique()

Out[53]: array(['UK', 'US', 'CA'], dtype=object)

In [54]: df3['intercept'] = 1
         df3[['UK', 'US', 'CA']] = pd.get_dummies(df3['country'])
         df3.drop('CA', axis = 1, inplace= True)

In [55]: df3.head()

Out[55]:         country                   timestamp      group landing_page  \
         user_id
         834778       UK  2017-01-14 23:08:43.304998    control     old_page
         928468       US  2017-01-23 14:44:16.387854  treatment     new_page
         822059       UK  2017-01-16 14:04:14.719771  treatment     new_page
         711597       UK  2017-01-22 03:14:24.763511    control     old_page
         710616       UK  2017-01-16 13:14:44.000513  treatment     new_page

                  converted  intercept  ab_page  UK  US
         user_id
         834778           0          1        0   0   1
         928468           0          1        1   0   0
         822059           1          1        1   0   1
         711597           0          1        0   0   1
         710616           0          1        1   0   1

In [56]: log= sm.Logit(df3['converted'], df3[['intercept','UK', 'US']])

In [57]: results= log.fit()

Optimization terminated successfully.
         Current function value: 0.366116
         Iterations 6


In [58]: results.summary2()

Out[58]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                  Results: Logit
         =================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2020-10-13 14:29 AIC:              212780.8333
         No. Observations:   290584           BIC:              212812.5723
         Df Model:           2                Log-Likelihood:   -1.0639e+05
         Df Residuals:       290581           LL-Null:          -1.0639e+05
         Converged:          1.0000           Scale:            1.0000
         -----------------------------------------------------------------
```

```
              Coef.    Std.Err.       z      P>|z|     [0.025    0.975]
------------------------------------------------------------------------
intercept    -1.9967     0.0068  -292.3145   0.0000   -2.0101   -1.9833
UK           -0.0408     0.0269    -1.5178   0.1291   -0.0935    0.0119
US            0.0099     0.0133     0.7458   0.4558   -0.0161    0.0360
========================================================================

"""
```

The logistic regression model indicated no significant effect between the independent variable (country) and the dependent variable (conversion). Country does not impact the conversion.

In [ ]:

In [ ]:

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 [59]:
```
df3['UK_newpage']= df3['UK'] * df3['ab_page']
df3['US_newpage']=df3['US'] * df3['ab_page']
```

In [60]:
```
df3.head()
```

Out[60]:

user_id	country	timestamp	group	landing_page	converted	intercept	ab_page	UK	US	UK_newpage	US_newpage
834778	UK	2017-01-14 23:08:43.304998	control	old_page	0	1	0	0	1	0	0
928468	US	2017-01-23 14:44:16.387854	treatment	new_page	0	1	1	0	0	0	0
822059	UK	2017-01-16 14:04:14.719771	treatment	new_page	1	1	1	0	1	0	1
711597	UK	2017-01-22 03:14:24.763511	control	old_page	0	1	0	0	1	0	0
710616	UK	2017-01-16 13:14:44.000513	treatment	new_page	0	1	1	0	1	0	1

In [61]:
```
log2= sm.Logit(df3['converted'], df3[['intercept','UK_newpage', 'US_newpage']])
```

In [62]:
```
res= log2. fit()
```

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

```
In [64]: res.summary2()

Out[64]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                  Results: Logit
         =================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2020-10-13 14:39 AIC:              212779.0384
         No. Observations:   290584           BIC:              212810.7773
         Df Model:           2                Log-Likelihood:   -1.0639e+05
         Df Residuals:       290581           LL-Null:          -1.0639e+05
         Converged:          1.0000           Scale:            1.0000
         -----------------------------------------------------------------
                       Coef.   Std.Err.      z      P>|z|    [0.025   0.975]
         -----------------------------------------------------------------
         intercept    -1.9963    0.0062  -322.0487  0.0000  -2.0084  -1.9841
         UK_newpage   -0.0752    0.0376    -1.9974  0.0458  -0.1489  -0.0014
         US_newpage    0.0149    0.0173     0.8617  0.3888  -0.0190   0.0488
         =================================================================

         """
```

When we run logistic regression for the group who received the new page and compared the countries, the UK had a significant effect (p= 0.045), but the US had an insignificant effect (p= 0.388) on conversion.

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

```
Out[75]: 0
```

```
In [ ]:
```