Analyze_ab_test_results_notebook

March 5, 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 seaborn as sb
        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')
```

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

```
In [3]: 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
```

c. The number of unique users in the dataset.

```
In [4]: len(df['user_id'].unique().tolist())
```

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

d. The proportion of users converted.

```
In [5]: df.converted.mean()

        #The proportion of users converted would be 11.9659%
```

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

```
In [8]: df.head()
```

```
Out[8]:    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
```

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

```
In [9]: df1 = df.loc[((df['group'] == 'treatment') & (df['landing_page'] != 'new_page'))]
        # Here I am setting group equals treatment and landing_page not equal to new_page
```

```
In [10]: df1.info()

         # From this we see that new_page and treatment do not match 1965 times
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1965 entries, 308 to 294252
Data columns (total 5 columns):
user_id         1965 non-null int64
timestamp       1965 non-null object
group           1965 non-null object
landing_page    1965 non-null object
converted       1965 non-null int64
dtypes: int64(2), object(3)
memory usage: 92.1+ KB
```

f. Do any of the rows have missing values?

```
In [11]: null_data = df[df.isnull().any(axis=1)]

In [12]: print(null_data)
```

```
Empty DataFrame
Columns: [user_id, timestamp, group, landing_page, converted]
Index: []
```

```
In [13]: df.dropna().shape[0]

Out[13]: 294478
```

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 [14]: df2 = df.loc[((df['group'] == 'treatment') & (df['landing_page'] == 'new_page')) | ((df

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

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

```
In [16]: # Double Check all of the correct rows were removed - this should be 0
         df3 = sum([((df2['group'] == 'treatment') & (df2['landing_page'] == 'old_page'))])

In [17]: df3.sum()
         # This returns 0 as is expected

Out[17]: 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 [18]: len(df2['user_id'].unique().tolist())

Out[18]: 290584

In [19]: len(df2['user_id'])

         # From this we do see that there is one duplicated user_id.

Out[19]: 290585
```

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

```
In [20]: df2[df2.duplicated(['user_id'], keep=False)].groupby(('user_id')).min()

Out[20]:                          timestamp      group landing_page  converted
         user_id
         773192   2017-01-09 05:37:58.781806  treatment     new_page          0
```

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

773192 2017-01-09 05:37:58.781806 treatment new_page 0

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

```
In [21]: df2 = df2.drop_duplicates(subset="user_id")

In [22]: len(df2['user_id'].unique().tolist())

Out[22]: 290584

In [23]: len(df2['user_id'])

         # Here we see that the 2 values are now the same showing that we did indeed remove the

Out[23]: 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 [24]: df['converted'].value_counts()

Out[24]: 0    259241
         1     35237
         Name: converted, dtype: int64
```

The probabillity of converting in this case is: 12%

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

```
In [25]: df2['group'].value_counts()

Out[25]: treatment    145310
         control      145274
         Name: group, dtype: int64

In [26]: df2_control = df2.loc[((df2['group'] == 'control') & (df2['converted'] == 1))]

In [27]: print(df2_control);

         # Here we see that the probabillity of conversion in this case is 17489/145274 which eq

        user_id                   timestamp    group landing_page  converted
4        864975  2017-01-21 01:52:26.210827  control     old_page          1
15       644214  2017-01-22 02:05:21.719434  control     old_page          1
28       913579  2017-01-24 09:11:39.164256  control     old_page          1
36       831737  2017-01-11 21:18:20.911015  control     old_page          1
43       862225  2017-01-08 14:49:37.335432  control     old_page          1
51       850231  2017-01-18 17:18:04.790584  control     old_page          1
67       834137  2017-01-24 12:20:07.638063  control     old_page          1
80       696202  2017-01-14 11:50:10.441615  control     old_page          1
89       832316  2017-01-18 16:58:44.742092  control     old_page          1
142      938283  2017-01-11 13:46:14.361746  control     old_page          1
145      857251  2017-01-14 15:28:49.816813  control     old_page          1
165      916276  2017-01-08 22:18:32.977546  control     old_page          1
178      862691  2017-01-07 13:49:55.346219  control     old_page          1
183      897941  2017-01-05 01:22:02.341061  control     old_page          1
193      724109  2017-01-19 15:40:41.197365  control     old_page          1
206      807785  2017-01-17 05:54:12.353372  control     old_page          1
210      939188  2017-01-19 20:37:23.918970  control     old_page          1
226      773693  2017-01-23 18:05:45.167335  control     old_page          1
255      861241  2017-01-16 07:29:31.485808  control     old_page          1
256      649981  2017-01-12 14:11:43.244243  control     old_page          1
261      853744  2017-01-07 17:26:40.733153  control     old_page          1
286      797845  2017-01-23 09:17:01.135951  control     old_page          1
289      723386  2017-01-22 13:15:48.814149  control     old_page          1
306      827871  2017-01-06 23:06:18.363087  control     old_page          1
347      792577  2017-01-07 01:57:04.651722  control     old_page          1
348      647872  2017-01-23 10:45:33.588479  control     old_page          1
```

```
378      689991  2017-01-19 12:33:28.343156  control     old_page          1
385      915326  2017-01-24 10:08:30.852573  control     old_page          1
451      807161  2017-01-15 20:54:18.343726  control     old_page          1
456      864258  2017-01-13 04:59:44.952951  control     old_page          1
...         ...                         ...      ...          ...        ...
293993   847479  2017-01-17 12:39:34.466178  control     old_page          1
294004   823691  2017-01-17 22:16:27.518085  control     old_page          1
294010   745649  2017-01-04 21:57:37.976078  control     old_page          1
294012   731522  2017-01-14 11:52:39.264688  control     old_page          1
294046   878870  2017-01-20 23:50:34.237011  control     old_page          1
294072   921479  2017-01-13 07:04:34.836390  control     old_page          1
294074   799201  2017-01-10 14:04:52.017189  control     old_page          1
294098   747241  2017-01-05 14:57:40.065394  control     old_page          1
294116   657168  2017-01-14 09:59:43.609646  control     old_page          1
294118   909676  2017-01-14 08:12:14.334756  control     old_page          1
294119   825715  2017-01-19 13:31:32.972679  control     old_page          1
294129   875197  2017-01-02 14:54:21.083435  control     old_page          1
294167   831624  2017-01-14 04:39:40.043173  control     old_page          1
294194   739723  2017-01-21 17:04:55.777070  control     old_page          1
294226   744272  2017-01-12 20:54:34.720360  control     old_page          1
294238   879861  2017-01-16 23:52:48.537037  control     old_page          1
294239   809271  2017-01-19 16:43:05.733777  control     old_page          1
294240   772692  2017-01-06 13:39:33.717153  control     old_page          1
294282   927192  2017-01-10 22:18:01.458159  control     old_page          1
294296   651092  2017-01-15 06:38:08.643136  control     old_page          1
294299   774203  2017-01-10 01:30:34.465084  control     old_page          1
294321   842711  2017-01-11 13:16:22.316188  control     old_page          1
294330   884352  2017-01-16 21:46:35.473427  control     old_page          1
294348   776042  2017-01-20 07:40:45.538528  control     old_page          1
294349   812331  2017-01-15 07:07:38.974256  control     old_page          1
294383   728029  2017-01-11 11:17:05.291493  control     old_page          1
294385   850065  2017-01-17 11:57:18.569907  control     old_page          1
294405   712217  2017-01-11 10:34:30.176801  control     old_page          1
294420   795742  2017-01-09 01:06:58.299207  control     old_page          1
294443   665217  2017-01-10 23:29:01.767720  control     old_page          1

[17489 rows x 5 columns]
```

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

```
In [28]: df2['group'].value_counts()

Out[28]: treatment    145310
         control      145274
         Name: group, dtype: int64

In [29]: new_df2 = df2.loc[((df2['group'] == 'treatment') & (df2['converted'] == 1))]
```

```
In [30]: print(new_df2);

         # Here we see that the probabillity of conversion in this case is 17264/145310 which eq
```

	user_id	timestamp	group	landing_page	converted
6	679687	2017-01-19 03:26:46.940749	treatment	new_page	1
8	817355	2017-01-04 17:58:08.979471	treatment	new_page	1
9	839785	2017-01-15 18:11:06.610965	treatment	new_page	1
17	888545	2017-01-08 06:37:26.332945	treatment	new_page	1
26	892356	2017-01-05 09:35:14.904865	treatment	new_page	1
32	875124	2017-01-05 15:39:25.439906	treatment	new_page	1
88	792499	2017-01-07 02:30:53.422520	treatment	new_page	1
106	800982	2017-01-13 18:37:56.692974	treatment	new_page	1
113	868502	2017-01-20 21:17:37.120529	treatment	new_page	1
117	704513	2017-01-20 08:42:33.420608	treatment	new_page	1
126	903977	2017-01-06 18:48:00.977241	treatment	new_page	1
135	732099	2017-01-23 19:55:46.763472	treatment	new_page	1
138	652015	2017-01-03 04:59:59.853514	treatment	new_page	1
149	803036	2017-01-18 16:11:42.188776	treatment	new_page	1
167	678160	2017-01-18 09:02:10.711673	treatment	new_page	1
186	716162	2017-01-20 13:23:57.095969	treatment	new_page	1
190	859483	2017-01-06 10:51:09.570850	treatment	new_page	1
192	656468	2017-01-18 07:13:29.805052	treatment	new_page	1
235	746858	2017-01-20 09:44:48.534145	treatment	new_page	1
238	680519	2017-01-10 07:06:41.561122	treatment	new_page	1
239	810338	2017-01-12 23:20:26.085000	treatment	new_page	1
253	662210	2017-01-08 04:17:28.613876	treatment	new_page	1
282	917603	2017-01-08 12:34:57.017073	treatment	new_page	1
284	839049	2017-01-17 08:20:29.229066	treatment	new_page	1
290	764241	2017-01-05 23:46:12.586146	treatment	new_page	1
297	736807	2017-01-20 19:12:24.514748	treatment	new_page	1
325	920908	2017-01-15 21:22:18.763101	treatment	new_page	1
355	817110	2017-01-11 00:33:06.846982	treatment	new_page	1
358	648312	2017-01-18 09:58:45.342868	treatment	new_page	1
360	757878	2017-01-23 22:01:31.530927	treatment	new_page	1
...	...	...	...	...	...
293837	804673	2017-01-04 18:09:26.845085	treatment	new_page	1
293863	756015	2017-01-03 20:20:27.235358	treatment	new_page	1
293936	709666	2017-01-23 20:11:00.202415	treatment	new_page	1
293955	755771	2017-01-11 09:36:16.241817	treatment	new_page	1
293963	824446	2017-01-09 17:13:06.473572	treatment	new_page	1
293967	727948	2017-01-12 22:47:19.403301	treatment	new_page	1
293979	694523	2017-01-03 21:03:07.062525	treatment	new_page	1
293982	780619	2017-01-07 17:35:00.987223	treatment	new_page	1
293990	742730	2017-01-02 21:56:12.177320	treatment	new_page	1
294000	773816	2017-01-12 01:12:12.947085	treatment	new_page	1
294023	644692	2017-01-08 19:37:44.913892	treatment	new_page	1
294024	830052	2017-01-08 19:45:26.905712	treatment	new_page	1

```
294041   821352  2017-01-18 02:15:41.229025  treatment     new_page          1
294056   771043  2017-01-06 07:13:58.128541  treatment     new_page          1
294161   664249  2017-01-15 15:40:52.660730  treatment     new_page          1
294175   783244  2017-01-07 14:31:16.189649  treatment     new_page          1
294185   889153  2017-01-02 13:43:35.689998  treatment     new_page          1
294195   855712  2017-01-22 11:24:24.240184  treatment     new_page          1
294202   939533  2017-01-21 04:16:33.930513  treatment     new_page          1
294280   700912  2017-01-21 15:42:40.293830  treatment     new_page          1
294285   667887  2017-01-07 22:25:37.059348  treatment     new_page          1
294301   771706  2017-01-13 04:07:24.601247  treatment     new_page          1
294319   703858  2017-01-17 10:28:05.059110  treatment     new_page          1
294338   760719  2017-01-18 03:56:22.321324  treatment     new_page          1
294358   815450  2017-01-18 02:40:09.551845  treatment     new_page          1
294371   653524  2017-01-13 19:30:59.784075  treatment     new_page          1
294382   919617  2017-01-23 09:02:44.935682  treatment     new_page          1
294388   757646  2017-01-14 08:55:00.303340  treatment     new_page          1
294396   838593  2017-01-15 09:56:31.455023  treatment     new_page          1
294430   733871  2017-01-21 17:54:08.810964  treatment     new_page          1

[17264 rows x 5 columns]
```

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

```
In [31]: df2['landing_page'].value_counts()

         # 145,310 users recieved the new_page out of 290,584. This means that 50% got the new p

Out[31]: new_page    145310
         old_page    145274
         Name: landing_page, dtype: int64
```

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.

11.96% of visitors converted regardless of which page they recived. 12.04% of visitors in the control group converted, while 11.88% of individuals in the treament group converted. These numbers are all so close that it shows that the page recived had no effect on conversions.

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.

My alternative Hypothesis is that the new page is better then the old page. My null hypothesis is that the new page is equal or worse then the old page

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 [32]: p_new = df2.query('(landing_page == "new_page")').converted.mean()

In [33]: p_new

Out[33]: 0.11880806551510564
```

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

```
In [34]: p_old = df2.query('(landing_page == "old_page")').converted.mean()

In [35]: p_old

Out[35]: 0.1203863045004612
```

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

```
In [36]: n_new = sum(df2.landing_page == 'new_page')

In [37]: print(n_new)

145310
```

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

```
In [38]: n_old = sum(df2.landing_page == 'old_page')

In [39]: print(n_old)

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 [40]:  new_converted_simulation = np.random.binomial(n_new, p_new, 10000)/n_new
          old_converted_simulation = np.random.binomial(n_old, p_old, 10000)/n_old
```

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).

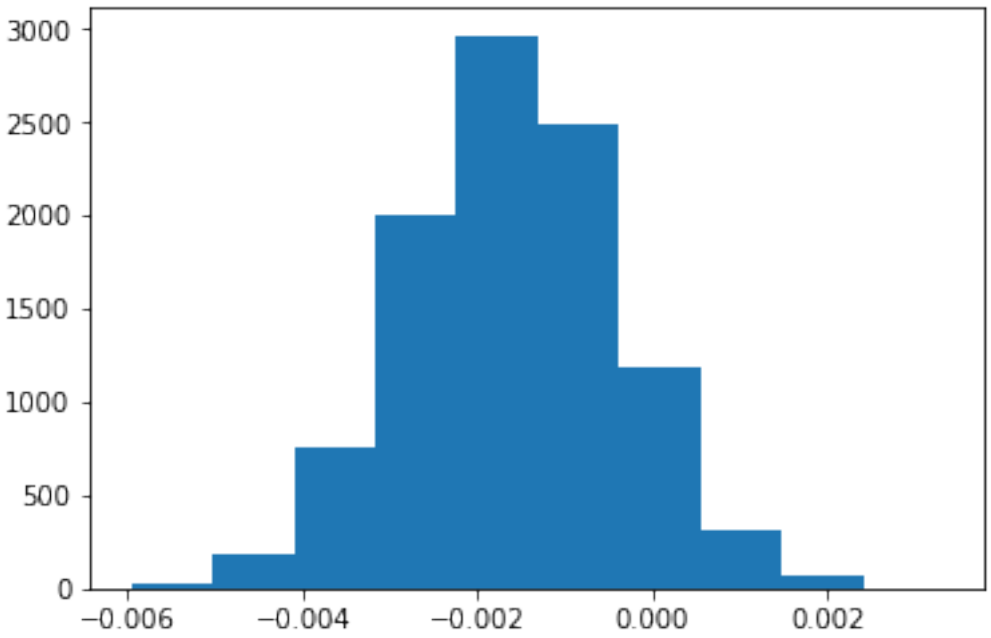
```
In [41]:  obs_diff = new_converted_simulation.mean() - old_converted_simulation.mean()

In [42]: obs_diff

Out[42]: -0.0016010986110203546
```

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 [43]: p_diffs = new_converted_simulation - old_converted_simulation
```

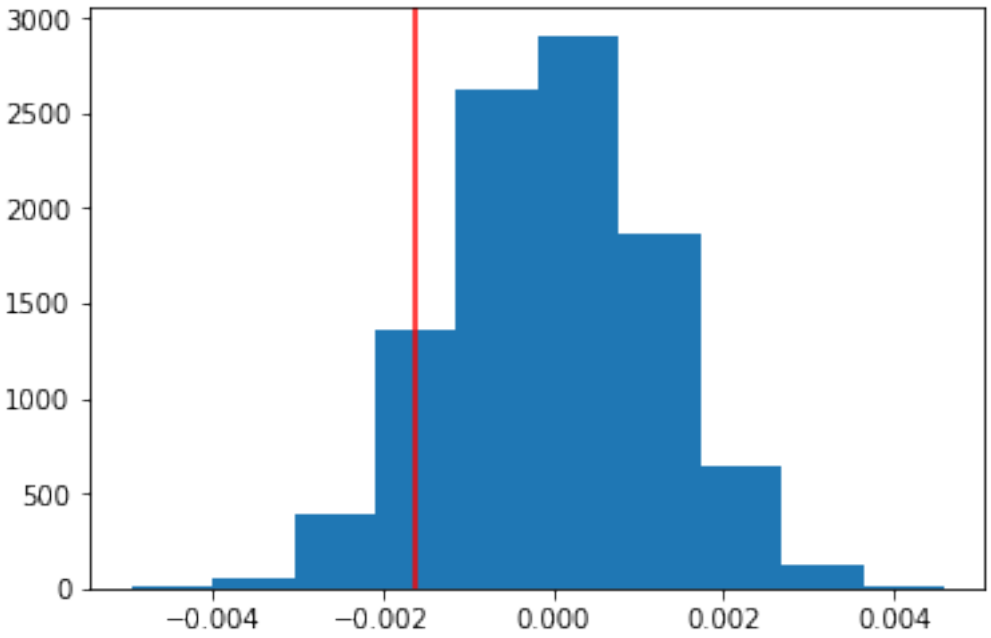
i. Plot a histogram of the **p_diffs**. Does this plot look like what you expected? Use the matching problem in the classroom to assure you fully understand what was computed here.

```
In [44]: plt.hist(p_diffs);

         # yes this graph does look like what I would expect based on the lesson in the classroo
```



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

```
In [45]: null_vals = np.random.normal(0, p_diffs.std(), p_diffs.size)
         plt.hist(null_vals)
         plt.axvline(obs_diff, c = 'red')

Out[45]: <matplotlib.lines.Line2D at 0x7f435dbbc4e0>
```




```
In [46]: (null_vals > obs_diff).mean()
         # From this we see that about 1/10 of values are greater then the actual difference we

Out[46]: 0.90490000000000004
```

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?

The red line that I found in part j is our observed statistic, what we are looking at here is if the p-value is greater then or equal to alpha or less then alpha. What we see from this is that values to the right of the line favor the alternative hypothesis. Due to our p-value being greater then our type 1 error rate we can reject the alternative and accept the null

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 [47]: df2.loc[((df2['landing_page'] == 'old_page') & (df2['converted'] == 1))];

In [48]: import statsmodels.api as sm

         convert_old = 17489
         convert_new = 17264
         n_old = 145274
         n_new = 145310;
```

```
/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 [49]: z_score, p_value = sm.stats.proportions_ztest([convert_new, convert_old], [n_new, n_old

In [50]: z_score

Out[50]: -1.3109241984234394

In [51]: p_value

Out[51]: 0.90505831275902449
```

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 z_score tells us how many standard deviations we are away from the mean. In this case we were about 1 and 1/3 std deveations to the left. Our p_value is very high in this case which tells me that our result is likely to be a false finding. **

Part III - A regression approach

** For the regression model in this section I will be using a logistic regression model.**

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 [52]: df2 = df2.assign(intercept = 1)

In [53]: print(df2);

        user_id                   timestamp      group landing_page  \
0        851104  2017-01-21 22:11:48.556739    control     old_page
1        804228  2017-01-12 08:01:45.159739    control     old_page
2        661590  2017-01-11 16:55:06.154213  treatment     new_page
```

3	853541	2017-01-08 18:28:03.143765	treatment	new_page
4	864975	2017-01-21 01:52:26.210827	control	old_page
5	936923	2017-01-10 15:20:49.083499	control	old_page
6	679687	2017-01-19 03:26:46.940749	treatment	new_page
7	719014	2017-01-17 01:48:29.539573	control	old_page
8	817355	2017-01-04 17:58:08.979471	treatment	new_page
9	839785	2017-01-15 18:11:06.610965	treatment	new_page
10	929503	2017-01-18 05:37:11.527370	treatment	new_page
11	834487	2017-01-21 22:37:47.774891	treatment	new_page
12	803683	2017-01-09 06:05:16.222706	treatment	new_page
13	944475	2017-01-22 01:31:09.573836	treatment	new_page
14	718956	2017-01-22 11:45:11.327945	treatment	new_page
15	644214	2017-01-22 02:05:21.719434	control	old_page
16	847721	2017-01-17 14:01:00.090575	control	old_page
17	888545	2017-01-08 06:37:26.332945	treatment	new_page
18	650559	2017-01-24 11:55:51.084801	control	old_page
19	935734	2017-01-17 20:33:37.428378	control	old_page
20	740805	2017-01-12 18:59:45.453277	treatment	new_page
21	759875	2017-01-09 16:11:58.806110	treatment	new_page
23	793849	2017-01-23 22:36:10.742811	treatment	new_page
24	905617	2017-01-20 14:12:19.345499	treatment	new_page
25	746742	2017-01-23 11:38:29.592148	control	old_page
26	892356	2017-01-05 09:35:14.904865	treatment	new_page
27	773302	2017-01-12 08:29:49.810594	treatment	new_page
28	913579	2017-01-24 09:11:39.164256	control	old_page
29	736159	2017-01-06 01:50:21.318242	treatment	new_page
30	690284	2017-01-13 17:22:57.182769	control	old_page
...	...	...	...	...
294448	776137	2017-01-12 05:53:12.386730	treatment	new_page
294449	883344	2017-01-22 23:15:58.645325	treatment	new_page
294450	825594	2017-01-06 12:37:08.897784	treatment	new_page
294451	875688	2017-01-14 07:19:49.042869	control	old_page
294452	927527	2017-01-12 10:52:11.084740	control	old_page
294453	789177	2017-01-17 18:17:56.215378	control	old_page
294454	937338	2017-01-19 03:23:22.236666	treatment	new_page
294455	733101	2017-01-23 12:52:58.711914	treatment	new_page
294456	679096	2017-01-02 16:43:49.237940	treatment	new_page
294457	691699	2017-01-09 23:42:35.963486	treatment	new_page
294458	807595	2017-01-22 10:43:09.285426	treatment	new_page
294459	924816	2017-01-20 10:59:03.481635	control	old_page
294460	846225	2017-01-16 15:24:46.705903	treatment	new_page
294461	740310	2017-01-10 17:22:19.762612	control	old_page
294462	677163	2017-01-03 19:41:51.902148	treatment	new_page
294463	832080	2017-01-19 13:18:27.352570	control	old_page
294464	834362	2017-01-17 01:51:56.106436	control	old_page
294465	925675	2017-01-07 20:38:26.346410	treatment	new_page
294466	923948	2017-01-09 16:33:41.104573	control	old_page
294467	857744	2017-01-05 08:00:56.024226	control	old_page

```
294468   643562  2017-01-02 19:20:05.460595  treatment     new_page
294469   755438  2017-01-18 17:35:06.149568    control     old_page
294470   908354  2017-01-11 02:42:21.195145    control     old_page
294471   718310  2017-01-21 22:44:20.378320    control     old_page
294472   822004  2017-01-04 03:36:46.071379  treatment     new_page
294473   751197  2017-01-03 22:28:38.630509    control     old_page
294474   945152  2017-01-12 00:51:57.078372    control     old_page
294475   734608  2017-01-22 11:45:03.439544    control     old_page
294476   697314  2017-01-15 01:20:28.957438    control     old_page
294477   715931  2017-01-16 12:40:24.467417  treatment     new_page
```

	converted	intercept
0	0	1
1	0	1
2	0	1
3	0	1
4	1	1
5	0	1
6	1	1
7	0	1
8	1	1
9	1	1
10	0	1
11	0	1
12	0	1
13	0	1
14	0	1
15	1	1
16	0	1
17	1	1
18	0	1
19	0	1
20	0	1
21	0	1
23	0	1
24	0	1
25	0	1
26	1	1
27	0	1
28	1	1
29	0	1
30	0	1
...	...	...
294448	0	1
294449	0	1
294450	0	1
294451	0	1
294452	0	1

```
294453          0          1
294454          0          1
294455          0          1
294456          0          1
294457          0          1
294458          0          1
294459          0          1
294460          0          1
294461          0          1
294462          0          1
294463          0          1
294464          0          1
294465          0          1
294466          0          1
294467          0          1
294468          0          1
294469          0          1
294470          0          1
294471          0          1
294472          0          1
294473          0          1
294474          0          1
294475          0          1
294476          0          1
294477          0          1

[290584 rows x 6 columns]
```

```
In [54]: df2 = df2.assign(ab_page = 1)
```

```
In [55]: print(df2)
```

```
        user_id                   timestamp      group landing_page  \
0        851104  2017-01-21 22:11:48.556739    control     old_page
1        804228  2017-01-12 08:01:45.159739    control     old_page
2        661590  2017-01-11 16:55:06.154213  treatment     new_page
3        853541  2017-01-08 18:28:03.143765  treatment     new_page
4        864975  2017-01-21 01:52:26.210827    control     old_page
5        936923  2017-01-10 15:20:49.083499    control     old_page
6        679687  2017-01-19 03:26:46.940749  treatment     new_page
7        719014  2017-01-17 01:48:29.539573    control     old_page
8        817355  2017-01-04 17:58:08.979471  treatment     new_page
9        839785  2017-01-15 18:11:06.610965  treatment     new_page
10       929503  2017-01-18 05:37:11.527370  treatment     new_page
11       834487  2017-01-21 22:37:47.774891  treatment     new_page
12       803683  2017-01-09 06:05:16.222706  treatment     new_page
13       944475  2017-01-22 01:31:09.573836  treatment     new_page
```

14	718956	2017-01-22 11:45:11.327945	treatment	new_page
15	644214	2017-01-22 02:05:21.719434	control	old_page
16	847721	2017-01-17 14:01:00.090575	control	old_page
17	888545	2017-01-08 06:37:26.332945	treatment	new_page
18	650559	2017-01-24 11:55:51.084801	control	old_page
19	935734	2017-01-17 20:33:37.428378	control	old_page
20	740805	2017-01-12 18:59:45.453277	treatment	new_page
21	759875	2017-01-09 16:11:58.806110	treatment	new_page
23	793849	2017-01-23 22:36:10.742811	treatment	new_page
24	905617	2017-01-20 14:12:19.345499	treatment	new_page
25	746742	2017-01-23 11:38:29.592148	control	old_page
26	892356	2017-01-05 09:35:14.904865	treatment	new_page
27	773302	2017-01-12 08:29:49.810594	treatment	new_page
28	913579	2017-01-24 09:11:39.164256	control	old_page
29	736159	2017-01-06 01:50:21.318242	treatment	new_page
30	690284	2017-01-13 17:22:57.182769	control	old_page
...	...	...	...	...
294448	776137	2017-01-12 05:53:12.386730	treatment	new_page
294449	883344	2017-01-22 23:15:58.645325	treatment	new_page
294450	825594	2017-01-06 12:37:08.897784	treatment	new_page
294451	875688	2017-01-14 07:19:49.042869	control	old_page
294452	927527	2017-01-12 10:52:11.084740	control	old_page
294453	789177	2017-01-17 18:17:56.215378	control	old_page
294454	937338	2017-01-19 03:23:22.236666	treatment	new_page
294455	733101	2017-01-23 12:52:58.711914	treatment	new_page
294456	679096	2017-01-02 16:43:49.237940	treatment	new_page
294457	691699	2017-01-09 23:42:35.963486	treatment	new_page
294458	807595	2017-01-22 10:43:09.285426	treatment	new_page
294459	924816	2017-01-20 10:59:03.481635	control	old_page
294460	846225	2017-01-16 15:24:46.705903	treatment	new_page
294461	740310	2017-01-10 17:22:19.762612	control	old_page
294462	677163	2017-01-03 19:41:51.902148	treatment	new_page
294463	832080	2017-01-19 13:18:27.352570	control	old_page
294464	834362	2017-01-17 01:51:56.106436	control	old_page
294465	925675	2017-01-07 20:38:26.346410	treatment	new_page
294466	923948	2017-01-09 16:33:41.104573	control	old_page
294467	857744	2017-01-05 08:00:56.024226	control	old_page
294468	643562	2017-01-02 19:20:05.460595	treatment	new_page
294469	755438	2017-01-18 17:35:06.149568	control	old_page
294470	908354	2017-01-11 02:42:21.195145	control	old_page
294471	718310	2017-01-21 22:44:20.378320	control	old_page
294472	822004	2017-01-04 03:36:46.071379	treatment	new_page
294473	751197	2017-01-03 22:28:38.630509	control	old_page
294474	945152	2017-01-12 00:51:57.078372	control	old_page
294475	734608	2017-01-22 11:45:03.439544	control	old_page
294476	697314	2017-01-15 01:20:28.957438	control	old_page
294477	715931	2017-01-16 12:40:24.467417	treatment	new_page

	converted	intercept	ab_page
0	0	1	1
1	0	1	1
2	0	1	1
3	0	1	1
4	1	1	1
5	0	1	1
6	1	1	1
7	0	1	1
8	1	1	1
9	1	1	1
10	0	1	1
11	0	1	1
12	0	1	1
13	0	1	1
14	0	1	1
15	1	1	1
16	0	1	1
17	1	1	1
18	0	1	1
19	0	1	1
20	0	1	1
21	0	1	1
23	0	1	1
24	0	1	1
25	0	1	1
26	1	1	1
27	0	1	1
28	1	1	1
29	0	1	1
30	0	1	1
...	...	...	...
294448	0	1	1
294449	0	1	1
294450	0	1	1
294451	0	1	1
294452	0	1	1
294453	0	1	1
294454	0	1	1
294455	0	1	1
294456	0	1	1
294457	0	1	1
294458	0	1	1
294459	0	1	1
294460	0	1	1
294461	0	1	1
294462	0	1	1
294463	0	1	1

```
294464          0          1        1
294465          0          1        1
294466          0          1        1
294467          0          1        1
294468          0          1        1
294469          0          1        1
294470          0          1        1
294471          0          1        1
294472          0          1        1
294473          0          1        1
294474          0          1        1
294475          0          1        1
294476          0          1        1
294477          0          1        1

[290584 rows x 7 columns]
```

```
In [56]: df2.loc[df2['group'] == 'control', 'ab_page'] = 1
         df2.loc[df2['group'] == 'treatment', 'ab_page'] = 0
```

```
In [57]: print(df2)
```

```
        user_id                   timestamp      group landing_page  \
0        851104  2017-01-21 22:11:48.556739    control     old_page
1        804228  2017-01-12 08:01:45.159739    control     old_page
2        661590  2017-01-11 16:55:06.154213  treatment     new_page
3        853541  2017-01-08 18:28:03.143765  treatment     new_page
4        864975  2017-01-21 01:52:26.210827    control     old_page
5        936923  2017-01-10 15:20:49.083499    control     old_page
6        679687  2017-01-19 03:26:46.940749  treatment     new_page
7        719014  2017-01-17 01:48:29.539573    control     old_page
8        817355  2017-01-04 17:58:08.979471  treatment     new_page
9        839785  2017-01-15 18:11:06.610965  treatment     new_page
10       929503  2017-01-18 05:37:11.527370  treatment     new_page
11       834487  2017-01-21 22:37:47.774891  treatment     new_page
12       803683  2017-01-09 06:05:16.222706  treatment     new_page
13       944475  2017-01-22 01:31:09.573836  treatment     new_page
14       718956  2017-01-22 11:45:11.327945  treatment     new_page
15       644214  2017-01-22 02:05:21.719434    control     old_page
16       847721  2017-01-17 14:01:00.090575    control     old_page
17       888545  2017-01-08 06:37:26.332945  treatment     new_page
18       650559  2017-01-24 11:55:51.084801    control     old_page
19       935734  2017-01-17 20:33:37.428378    control     old_page
20       740805  2017-01-12 18:59:45.453277  treatment     new_page
21       759875  2017-01-09 16:11:58.806110  treatment     new_page
23       793849  2017-01-23 22:36:10.742811  treatment     new_page
24       905617  2017-01-20 14:12:19.345499  treatment     new_page
```

```
25       746742  2017-01-23 11:38:29.592148    control     old_page
26       892356  2017-01-05 09:35:14.904865  treatment     new_page
27       773302  2017-01-12 08:29:49.810594  treatment     new_page
28       913579  2017-01-24 09:11:39.164256    control     old_page
29       736159  2017-01-06 01:50:21.318242  treatment     new_page
30       690284  2017-01-13 17:22:57.182769    control     old_page
...         ...                         ...        ...          ...
294448   776137  2017-01-12 05:53:12.386730  treatment     new_page
294449   883344  2017-01-22 23:15:58.645325  treatment     new_page
294450   825594  2017-01-06 12:37:08.897784  treatment     new_page
294451   875688  2017-01-14 07:19:49.042869    control     old_page
294452   927527  2017-01-12 10:52:11.084740    control     old_page
294453   789177  2017-01-17 18:17:56.215378    control     old_page
294454   937338  2017-01-19 03:23:22.236666  treatment     new_page
294455   733101  2017-01-23 12:52:58.711914  treatment     new_page
294456   679096  2017-01-02 16:43:49.237940  treatment     new_page
294457   691699  2017-01-09 23:42:35.963486  treatment     new_page
294458   807595  2017-01-22 10:43:09.285426  treatment     new_page
294459   924816  2017-01-20 10:59:03.481635    control     old_page
294460   846225  2017-01-16 15:24:46.705903  treatment     new_page
294461   740310  2017-01-10 17:22:19.762612    control     old_page
294462   677163  2017-01-03 19:41:51.902148  treatment     new_page
294463   832080  2017-01-19 13:18:27.352570    control     old_page
294464   834362  2017-01-17 01:51:56.106436    control     old_page
294465   925675  2017-01-07 20:38:26.346410  treatment     new_page
294466   923948  2017-01-09 16:33:41.104573    control     old_page
294467   857744  2017-01-05 08:00:56.024226    control     old_page
294468   643562  2017-01-02 19:20:05.460595  treatment     new_page
294469   755438  2017-01-18 17:35:06.149568    control     old_page
294470   908354  2017-01-11 02:42:21.195145    control     old_page
294471   718310  2017-01-21 22:44:20.378320    control     old_page
294472   822004  2017-01-04 03:36:46.071379  treatment     new_page
294473   751197  2017-01-03 22:28:38.630509    control     old_page
294474   945152  2017-01-12 00:51:57.078372    control     old_page
294475   734608  2017-01-22 11:45:03.439544    control     old_page
294476   697314  2017-01-15 01:20:28.957438    control     old_page
294477   715931  2017-01-16 12:40:24.467417  treatment     new_page

        converted  intercept  ab_page
0               0          1        1
1               0          1        1
2               0          1        0
3               0          1        0
4               1          1        1
5               0          1        1
6               1          1        0
7               0          1        1
8               1          1        0
```

9	1	1	0
10	0	1	0
11	0	1	0
12	0	1	0
13	0	1	0
14	0	1	0
15	1	1	1
16	0	1	1
17	1	1	0
18	0	1	1
19	0	1	1
20	0	1	0
21	0	1	0
23	0	1	0
24	0	1	0
25	0	1	1
26	1	1	0
27	0	1	0
28	1	1	1
29	0	1	0
30	0	1	1
...	...	...	...
294448	0	1	0
294449	0	1	0
294450	0	1	0
294451	0	1	1
294452	0	1	1
294453	0	1	1
294454	0	1	0
294455	0	1	0
294456	0	1	0
294457	0	1	0
294458	0	1	0
294459	0	1	1
294460	0	1	0
294461	0	1	1
294462	0	1	0
294463	0	1	1
294464	0	1	1
294465	0	1	0
294466	0	1	1
294467	0	1	1
294468	0	1	0
294469	0	1	1
294470	0	1	1
294471	0	1	1
294472	0	1	0
294473	0	1	1

```
294474          0          1        1
294475          0          1        1
294476          0          1        1
294477          0          1        0

[290584 rows x 7 columns]
```

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 [62]: lm = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])
         results = lm.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 [63]: results.summary2()

Out[63]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                 Results: Logit
         =================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2020-03-05 15:51 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    -2.0038    0.0081  -247.1457  0.0000  -2.0197  -1.9879
         ab_page       0.0150    0.0114     1.3109  0.1899  -0.0074   0.0374
         =================================================================

         """
```

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**?

We are looking at the likelyhood of conversion based on if an individual was in either the treatment group or the control group. The p-value for ab_page is 0.1899 this value is less then our threshold which indicates strong evidence against the null hypotesis.

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 main reason that adding more terms is beneficial is that there could be a relationship that is affecting your outcome and you might not even see it unless you have the right combination of terms. One big disabvantage of adding more terms is that more terms means more complexity and the more complex your problem is the more likely you are to make a mistake.

g. Does it appear that country had an impact on conversion?.

```
In [65]: df_c = pd.read_csv('countries.csv')

In [66]: df_c.info()
         df_c.country.unique()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 290584 entries, 0 to 290583
Data columns (total 2 columns):
user_id    290584 non-null int64
country    290584 non-null object
dtypes: int64(1), object(1)
memory usage: 4.4+ MB


Out[66]: array(['UK', 'US', 'CA'], dtype=object)

In [67]: df_c[['A', 'B', 'C']] = pd.get_dummies(df_c['country'])

In [68]: df_c = df_c.drop('C', 1);

In [69]: df_c.head()

Out[69]:    user_id country  A  B
         0   834778      UK  0  1
         1   928468      US  0  0
         2   822059      UK  0  1
         3   711597      UK  0  1
         4   710616      UK  0  1

In [70]: df_final = pd.merge(df2, df_c, on='user_id')

In [71]: df_final.head()
```

```
Out[71]:    user_id                   timestamp      group landing_page  converted  \
         0   851104  2017-01-21 22:11:48.556739    control     old_page          0
         1   804228  2017-01-12 08:01:45.159739    control     old_page          0
         2   661590  2017-01-11 16:55:06.154213  treatment     new_page          0
         3   853541  2017-01-08 18:28:03.143765  treatment     new_page          0
         4   864975  2017-01-21 01:52:26.210827    control     old_page          1

            intercept  ab_page country  A  B
         0          1        1      US  0  0
         1          1        1      US  0  0
         2          1        0      US  0  0
         3          1        0      US  0  0
         4          1        1      US  0  0
```

```
In [72]: lm_new = sm.Logit(df_final['converted'], df_final[['intercept', 'A', 'B']])
         new_results = lm_new.fit()

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

```
In [73]: new_results.summary2()
```

```
Out[73]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                 Results: Logit
         =================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2020-03-05 15:52 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
         A            -0.0408    0.0269    -1.5178  0.1291  -0.0935   0.0119
         B             0.0099    0.0133     0.7458  0.4558  -0.0161   0.0360
         =================================================================

         """
```

There does seem to be a correlation between country and conversion. We see that the p-value for country B, which was the UK, was almost 4 times greater then the p-value for country A, which was 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 [78]: df_final[['page_A', 'page_B']] = pd.get_dummies(df_final['landing_page'])

In [79]: df_final.head()
```

Out[79]:

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

```
In [81]: lm_new2 = sm.Logit(df_final['converted'], df_final[['intercept', 'page_A', 'page_B']])
         new_results = lm_new2.fit()

Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.366118
         Iterations: 35


/opt/conda/lib/python3.6/site-packages/statsmodels/base/model.py:496: ConvergenceWarning: Maximu
  "Check mle_retvals", ConvergenceWarning)


In [83]: new_results.summary2()

Out[83]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                  Results: Logit
         ================================================================
         Model:              Logit            No. Iterations:    35.0000
         Dependent Variable: converted        Pseudo R-squared:  0.000
         Date:               2020-03-05 15:58 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:          0.0000           Scale:             1.0000
```

```
------------------------------------------------------------------------------
           Coef.    Std.Err.      z    P>|z|     [0.025       0.975]
------------------------------------------------------------------------------
intercept -1.3308 126484.7541 -0.0000 1.0000 -247906.8934 247904.2317
page_A    -0.6729 126484.7541 -0.0000 1.0000 -247906.2355 247904.8896
page_B    -0.6579 126484.7541 -0.0000 1.0000 -247906.2205 247904.9046
==============================================================================

"""
```

We get a final p-value of 1 which is greater then the type 1 error rate and so we can reject the alternative and accept the null.

```
In [84]: from subprocess import call
         call(['python', '-m', 'nbconvert', 'Analyze_ab_test_results_notebook.ipynb'])
```

```
Out[84]: 0
```

```
In [ ]:
```