Analyze_ab_test_results_notebook

June 18, 2022

1 Analyze A/B Test Results

This project will assure you have mastered the subjects covered in the statistics lessons. We have organized the current notebook into the following sections:

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

Specific programming tasks are marked with a **ToDo** tag.

Introduction

A/B tests are very commonly performed by data analysts and data scientists. 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 webpage, - Keep the old webpage, or - Perhaps run the experiment longer to make their decision.

Each **ToDo** task below has an associated quiz present in the classroom. Though the classroom quizzes are **not necessary** to complete the project, they help ensure you are on the right track as you work through the project, and you can feel more confident in your final submission meeting the rubric specification.

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)
```

1.0.1 ToDo 1.1

Now, read in the `ab_data.csv` data. Store it in `df`. Below is the description of the data, there are a total of 5 columns:

Data columns	Purpose	Valid values
user_id	Unique ID	Int64 values
timestamp	Time stamp when the user visited the webpage	-
group	In the current A/B experiment, the users are categorized into two broad groups. The `control` group users are expected to be served with `old_page`; and `treatment` group users are matched with the `new_page`. However, **some inaccurate rows** are present in the initial data, such as a `control` group user is matched with a `new_page`.	`['control', 'treatment']`
landing_page	It denotes whether the user visited the old or new webpage.	`['old_page', 'new_page']`
converted	It denotes whether the user decided to pay for the company's product. Here, `1` means yes, the user bought the product.	`[0, 1]`

Use your dataframe to answer the questions in Quiz 1 of the classroom.
a. Read in the dataset from the `ab_data.csv` file 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[0]
```

```
Out[3]: 294478
```

c. The number of unique users in the dataset.

```
In [4]: df.user_id.nunique()
```

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

d. The proportion of users converted.

```
In [5]: (df.converted == 1).mean()
```

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

e. The number of times when the "group" is `treatment` but "landing_page" is not a `new_page`.

```
In [6]: df.query("(group == 'treatment' and landing_page == 'old_page')").shape[0]
```

```
Out[6]: 1965
```

f. Do any of the rows have missing values?

```
In [7]: df.isnull().values.any()
```

```
Out[7]: False
```

1.0.2 ToDo 1.2

In a particular row, the **group** and **landing_page** columns should have either of the following acceptable values:

user_id	timestamp	group	landing_page	converted
XXXX	XXXX	`control`	`old_page`	X
XXXX	XXXX	`treatment`	`new_page`	X

It means, the `control` group users should match with `old_page`; and `treatment` group users should matched with the `new_page`.

However, for the rows where `treatment` does not match with `new_page` or `control` does not

match with old_page, we cannot be sure if such rows truly received the new or old wepage.

Use **Quiz 2** in the classroom to figure out how should we handle the rows where the group and landing_page columns don't match?

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.drop(df.query('(group == "treatment" and landing_page != "new_page") or (group
        df2.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

```
In [9]: # Double Check all of the incorrect rows were removed from df2 -
        # Output of the statement below should be 0
        df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].sha
```

```
Out[9]: 0
```

```
In [10]: # another check  of the incorrect rows were removed from df2 -
         # Output of the statement below should be 0
         df2[((df2['group'] == 'control') == (df2['landing_page'] == 'old_page')) == False].shap
```

```
Out[10]: 0
```

1.0.3 ToDo 1.3

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

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

```
In [11]: df2.user_id.nunique()
```

```
Out[11]: 290584
```

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

```
In [12]: df2[df2.duplicated(['user_id'])]
```

Out[12]:

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

d. Remove **one** of the rows with a duplicate **user_id**, from the **df2** dataframe.

```
In [14]: # Remove one of the rows with a duplicate user_id..
         # Hint: The dataframe.drop_duplicates() may not work in this case because the rows with
         # Check again if the row with a duplicate user_id is deleted or not
         df2.drop_duplicates(['user_id'])
```

Out[14]:

	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
20	740805	2017-01-12 18:59:45.453277	treatment	new_page	0
21	759875	2017-01-09 16:11:58.806110	treatment	new_page	0
23	793849	2017-01-23 22:36:10.742811	treatment	new_page	0
24	905617	2017-01-20 14:12:19.345499	treatment	new_page	0
25	746742	2017-01-23 11:38:29.592148	control	old_page	0
26	892356	2017-01-05 09:35:14.904865	treatment	new_page	1
27	773302	2017-01-12 08:29:49.810594	treatment	new_page	0
28	913579	2017-01-24 09:11:39.164256	control	old_page	1
29	736159	2017-01-06 01:50:21.318242	treatment	new_page	0
30	690284	2017-01-13 17:22:57.182769	control	old_page	0
...	...	...	...	...	...
294448	776137	2017-01-12 05:53:12.386730	treatment	new_page	0
294449	883344	2017-01-22 23:15:58.645325	treatment	new_page	0
294450	825594	2017-01-06 12:37:08.897784	treatment	new_page	0
294451	875688	2017-01-14 07:19:49.042869	control	old_page	0
294452	927527	2017-01-12 10:52:11.084740	control	old_page	0
294453	789177	2017-01-17 18:17:56.215378	control	old_page	0
294454	937338	2017-01-19 03:23:22.236666	treatment	new_page	0
294455	733101	2017-01-23 12:52:58.711914	treatment	new_page	0
294456	679096	2017-01-02 16:43:49.237940	treatment	new_page	0
294457	691699	2017-01-09 23:42:35.963486	treatment	new_page	0
294458	807595	2017-01-22 10:43:09.285426	treatment	new_page	0
294459	924816	2017-01-20 10:59:03.481635	control	old_page	0
294460	846225	2017-01-16 15:24:46.705903	treatment	new_page	0
294461	740310	2017-01-10 17:22:19.762612	control	old_page	0
294462	677163	2017-01-03 19:41:51.902148	treatment	new_page	0
294463	832080	2017-01-19 13:18:27.352570	control	old_page	0

```
294464   834362  2017-01-17 01:51:56.106436    control     old_page          0
294465   925675  2017-01-07 20:38:26.346410  treatment     new_page          0
294466   923948  2017-01-09 16:33:41.104573    control     old_page          0
294467   857744  2017-01-05 08:00:56.024226    control     old_page          0
294468   643562  2017-01-02 19:20:05.460595  treatment     new_page          0
294469   755438  2017-01-18 17:35:06.149568    control     old_page          0
294470   908354  2017-01-11 02:42:21.195145    control     old_page          0
294471   718310  2017-01-21 22:44:20.378320    control     old_page          0
294472   822004  2017-01-04 03:36:46.071379  treatment     new_page          0
294473   751197  2017-01-03 22:28:38.630509    control     old_page          0
294474   945152  2017-01-12 00:51:57.078372    control     old_page          0
294475   734608  2017-01-22 11:45:03.439544    control     old_page          0
294476   697314  2017-01-15 01:20:28.957438    control     old_page          0
294477   715931  2017-01-16 12:40:24.467417  treatment     new_page          0

[290584 rows x 5 columns]
```

1.0.4 ToDo 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 [15]: p_population = df2.query('converted == 1').user_id.nunique() / df2.shape[0]

In [16]: print('Probability of an individual converting regardless of the page they receive :',p

Probability of an individual converting regardless of the page they receive : 0.11959667567149022
```

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

```
In [17]: p_control = df2.query('group == "control" and converted == 1').user_id.nunique()/ df2.q

In [18]: print('Probability of converted in control group: ',p_control)

Probability of converted in control group:  0.1203863045004612
```

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

```
In [19]: p_treatment = df2.query('group == "treatment" and converted == 1').user_id.nunique()/ d

In [20]: print('Probability of converted in treatment group: ',p_treatment)

Probability of converted in treatment group:  0.11880806551510564
```

```
In [21]: # Calculate the actual difference (obs_diff) between the conversion rates for the two g
         obs_diff= p_treatment - p_control
         print('actual difference between the conversion rates for the treatment and control gro
```

```
actual difference between the conversion rates for the treatment and control groups: -0.00157823
```

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

```
In [22]: p_new_page=df2.query('landing_page == "new_page"').user_id.nunique()/ df2.shape[0]
         print('The probability of receiving the new page:',p_new_page)
```

```
The probability of receiving the new page: 0.5000602233425676
```

e. Consider your results from parts (a) through (d) above, and explain below whether the new `treatment` group users lead to more conversions.

> **As the probability converted rate in control group is larger than the probability of conversion rate of the treatment group , there is no evidence that the new page will lead to more conversion rate.**

Part II - A/B Test

Since a timestamp is associated with each event, you could run a hypothesis test continuously as long as you observe the events.

However, then the hard questions would be: - 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.

1.0.5 ToDo 2.1

For now, consider you need to make the decision just based on all the data provided.

> Recall that you just calculated that the "converted" probability (or rate) for the old page is *slightly* higher than that of the new page (ToDo 1.4.c).

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 be your null and alternative hypotheses (H_0 and H_1)?

You can state your hypothesis in terms of words or in terms of p_{old} and p_{new}, which are the "converted" probability (or rate) for the old and new pages respectively.

0 : <=

1 : >

1.0.6 ToDo 2.2 - Null Hypothesis H_0 Testing

Under the null hypothesis H_0, assume that p_{new} and p_{old} are equal. Furthermore, assume that p_{new} and p_{old} both are equal to the **converted** success rate in the `df2` data regardless of the page. So, our assumption is:

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

In this section, you will:

- Simulate (bootstrap) sample data set for both groups, and compute the "converted" probability p for those samples.
- Use a sample size for each group equal to the ones in the `df2` data.
- Compute the difference in the "converted" probability for the two samples above.
- Perform the sampling distribution for the "difference in the converted probability" between the two simulated-samples over 10,000 iterations; and calculate an estimate.

Use the cells below to provide the necessary parts of this simulation. 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 hypothesis?

```
In [23]: p_new= df2.query('converted == 1').user_id.nunique() / df2.shape[0]
         p_new

Out[23]: 0.11959667567149027
```

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

```
In [24]: p_old =p_new
         p_old

Out[24]: 0.11959667567149027
```

c. What is n_{new}, the number of individuals in the treatment group? *Hint*: The treatment group users are shown the new page.

```
In [25]: n_new=df2.query('landing_page == "new_page"').user_id.count()
         n_new

Out[25]: 145311
```

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

```
In [26]: n_old=df2.query('landing_page == "old_page"').user_id.count()
         n_old

Out[26]: 145274
```

e. Simulate Sample for the `treatment` Group Simulate n_{new} transactions with a conversion rate of p_{new} under the null hypothesis.

```
In [27]: # Simulate a Sample for the treatment Group
         new_page_converted=np.random.choice([1,0], size=n_new, p=[p_new,(1-p_new)])
```

f. Simulate Sample for the `control` Group Simulate n_{old} transactions with a conversion rate of p_{old} under the null hypothesis. Store these n_{old} 1's and 0's in the `old_page_converted` numpy array.

```
In [28]: # Simulate a Sample for the control Group
         old_page_converted=np.random.choice([1,0], size=n_old, p=[p_old, (1-p_old)])
```

g. Find the difference in the "converted" probability $(p'_{new}$ - $p'_{old})$ for your simulated samples from the parts (e) and (f) above.

```
In [30]: new_page_converted.mean()

Out[30]: 0.11895864731506905

In [31]: old_page_converted.mean()

Out[31]: 0.1185965830086595

In [32]: obs_diff = new_page_converted.mean() - old_page_converted.mean()
         print('The difference in the converted:',obs_diff)

The difference in the converted: 0.00036206430641
```

h. Sampling distribution Re-create `new_page_converted` and `old_page_converted` and find the $(p'_{new}$ - $p'_{old})$ value 10,000 times using the same simulation process you used in parts (a) through (g) above.

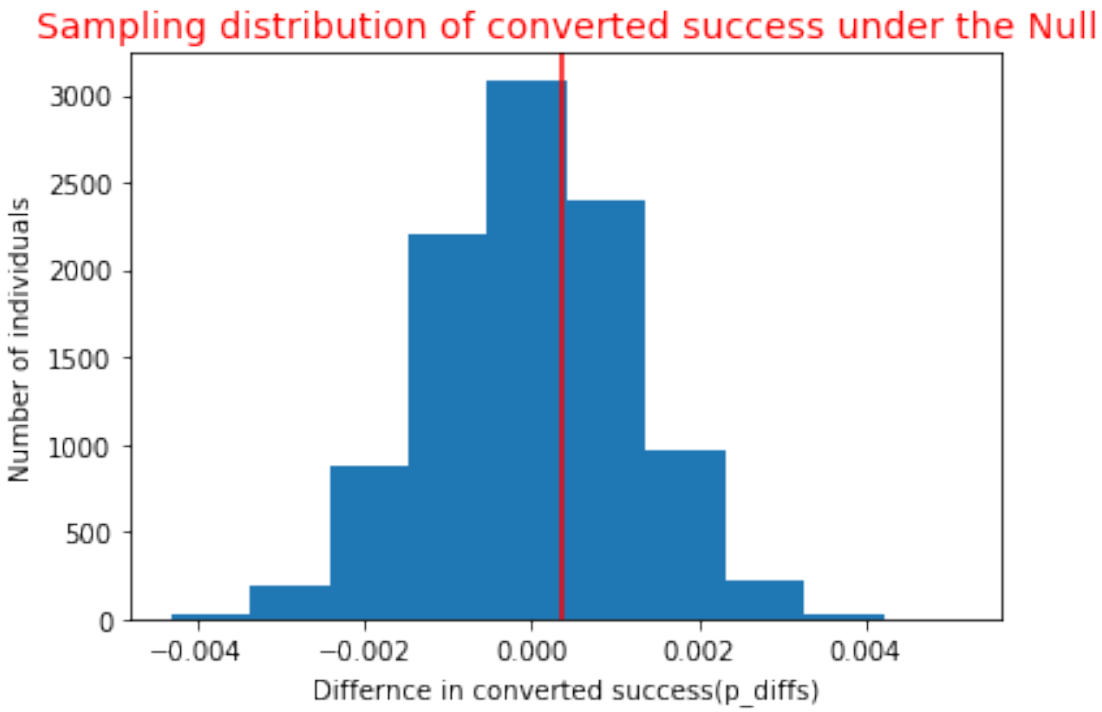
Store all $(p'_{new}$ - $p'_{old})$ values in a NumPy array called `p_diffs`.

```
In [33]: # Sampling distribution
         p_diffs = []
         new_page_converted = np.random.binomial(n_new, p_new, 10000)/n_new
         old_page_converted = np.random.binomial(n_old, p_old, 10000)/n_old
         p_diffs=new_page_converted - old_page_converted
```

i. Histogram 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.

Also, use `plt.axvline()` method to mark the actual difference observed in the `df2` data (recall `obs_diff`), in the chart.

```
In [34]: plt.hist(p_diffs);
         plt.axvline(obs_diff, color='red');
         plt.title("Sampling distribution of converted success under the Null", color='red',font
         plt.xlabel('Differnce in converted success(p_diffs)')
         plt.ylabel('Number of individuals');
```



j. What proportion of the **p_diffs** are greater than the actual difference observed in the `df2` data?

```
In [35]: obs_diff = df2[df2['group'] == 'treatment']['converted'].mean() -  df2[df2['group'] ==
         print(obs_diff)
         p_diffs = np.array(p_diffs)
         (obs_diff < p_diffs).mean()
```

```
-0.00157905659769
```

```
Out[35]: 0.90629999999999999
```

k. Please explain in words what you have just computed in part **j** above.
- What is this value called in scientific studies?
- What does this value signify in terms of whether or not there is a difference between the new and old pages? *Hint*: Compare the value above with the "Type I error rate (0.05)".

This value is called P_value

As P_value is greater than the actual difference in df2 ,we fail to reject the Null hypothesis as we we don't have evidance than new page conversion is higher than the old page

l. Using Built-in Methods for Hypothesis Testing 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 walk-through of the ideas that are critical to correctly thinking about statistical significance.

Fill in the statements below to calculate the: - `convert_old`: number of conversions with the old_page - `convert_new`: number of conversions with the new_page - `n_old`: number of individuals who were shown the old_page - `n_new`: number of individuals who were shown the new_page

```
In [36]: import statsmodels.api as sm

         # number of conversions with the old_page
         convert_old = df2.query('landing_page == "old_page" & converted== 1').user_id.count()

         # number of conversions with the new_page
         convert_new =df2.query('landing_page == "new_page" & converted== 1').user_id.count()

         # number of individuals who were shown the old_page
         n_old = df2.query('landing_page == "old_page"').user_id.count()

         # number of individuals who received new_page
         n_new = df2.query('landing_page == "new_page"').user_id.count()
```

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

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

The syntax is:

```
proportions_ztest(count_array, nobs_array, alternative='larger')
```

where, - `count_array` = represents the number of "converted" for each group - `nobs_array` = represents the total number of observations (rows) in each group - `alternative` = choose one of the values from `[two-sided, smaller, larger]` depending upon two-tailed, left-tailed, or right-tailed respectively.

The built-in function above will return the z_score, p_value.

```
In [37]: import statsmodels.api as sm
         # ToDo: Complete the sm.stats.proportions_ztest() method arguments
         z_score, p_value = sm.stats.proportions_ztest([convert_new, convert_old], [n_new, n_old
         print(z_score, p_value)
```

```
-1.31160753391 0.905173705141
```

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.**?

P_value and Z- score agree with the findinds we get earlier.

With P_value 0.91 , we fail to reject the Null Hypothesis and that mean we can't be confident that the conversion rate of the new page is higher than the conversion rate of the old page.

Part III - A regression approach

1.0.7 ToDo 3.1

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 in the `df2` data is either a conversion or no conversion, what type of regression should you be performing in this case?

> **As the type of data is categorical (conversion or no conversion), we will use logistic regression.**

b. The goal is to use **statsmodels** library to fit the regression model you specified in part **a.** above to see if there is a significant difference in conversion based on the page-type a customer receives. However, you first need to create the following two columns in the `df2` dataframe: 1. `intercept` - It should be `1` in the entire column. 2. `ab_page` - It's a dummy variable column, having a value `1` when an individual receives the **treatment**, otherwise `0`.

```
In [38]: #dummy variables

         df2[['control','treatment']] = pd.get_dummies(df['group'])
         df2=df2.drop('control', axis = 1)
         df3 = df2.rename(columns={'treatment': 'ab_page'})
         df3['intercept'] = 1
         df3.head()
```

Out[38]:

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

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

```
In [39]: log_mod = sm.Logit(df3['converted'],df3[['intercept','ab_page']])
         results = log_mod.fit()

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

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

```
In [41]: results.summary2()

Out[41]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                 Results: Logit
         =================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2022-06-18 19:14 AIC:              212780.6032
         No. Observations:   290585           BIC:              212801.7625
         Df Model:           1                Log-Likelihood:   -1.0639e+05
         Df Residuals:       290583           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    -1.3116  0.1897  -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**?

With P_value of ab_page 0.19 suggests that it is not statistically significant in predicting if the new page is highly converted or not.

P_value of ab_page is differ from in A/B test part may be because of assuming intercept.

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?

Advantages of considering other factors to add into regression model to get a clear a more precicious model on conversion rate.

Disadvantages to adding additional terms into regression model adding other factors will make the model more complex and one of the problems that could arise by considering other additional factors may be multicollinearity.

g. Adding countries Now along with testing if the conversion rate changes for different pages, also add an effect based on which country a user lives in.

1. You will need to read in the **countries.csv** dataset and merge together your `df2` datasets on the appropriate rows. You call the resulting dataframe `df_merged`. Here are the docs for joining tables.

2. Does it appear that country had an impact on conversion? To answer this question, consider the three unique values, `['UK', 'US', 'CA']`, in the `country` column. Create dummy variables for these country columns.

Provide the statistical output as well as a written response to answer this question.

```
In [43]: # Read the countries.csv
         countries_df=pd.read_csv('countries.csv')
         countries_df.head()
```

```
Out[43]:    user_id country
         0   834778      UK
         1   928468      US
         2   822059      UK
         3   711597      UK
         4   710616      UK
```

```
In [44]: # Join with the df2 dataframe
         df_merged=df3.join(countries_df.set_index('user_id'), on='user_id')
         df_merged.head()
```

```
Out[44]:    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 country
         0        0          1      US
         1        0          1      US
         2        1          1      US
         3        1          1      US
         4        0          1      US
```

```
In [45]: # Create the necessary dummy variables
         df_merged[['UK', 'US', 'CA']]=pd.get_dummies(df_merged['country'])
         df_merged.head()
```

```
Out[45]:    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 country  UK  US  CA
         0        0          1      US   0   0   1
         1        0          1      US   0   0   1
```

```
2        1          1      US   0   0   1
3        1          1      US   0   0   1
4        0          1      US   0   0   1
```

```
In [46]: df_merged['intercept'] = 1
         log_mod = sm.Logit(df_merged['converted'],df_merged[['intercept','ab_page','UK','CA']])
         results = log_mod.fit()
         results.summary2()
```

```
Optimization terminated successfully.
         Current function value: 0.366112
         Iterations 6
```

```
Out[46]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                Results: Logit
         =================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2022-06-18 19:16 AIC:              212781.3782
         No. Observations:   290585           BIC:              212823.6968
         Df Model:           3                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    -1.9794    0.0127  -155.4143  0.0000  -2.0043  -1.9544
         ab_page      -0.0150    0.0114    -1.3076  0.1910  -0.0374   0.0075
         UK           -0.0506    0.0284    -1.7835  0.0745  -0.1063   0.0050
         CA           -0.0099    0.0133    -0.7437  0.4570  -0.0359   0.0162
         =================================================================

         """
```

h. Fit your model and obtain the results 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 are there significant effects on conversion. **Create the necessary additional columns, and fit the new model.**

Provide the summary results (statistical output), and your conclusions (written response) based on the results.

```
In [49]: # Fit your model, and summarize the results
         df_merged['ab_UK'] = df_merged['ab_page'] * df_merged['UK']
         df_merged['ab_CA'] = df_merged['ab_page'] * df_merged['CA']
         logit_mod = sm.Logit(df_merged['converted'], df_merged[['intercept','ab_page','UK','CA'
         results = logit_mod.fit()
         results.summary2()
```

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

Out[49]:
```
<class 'statsmodels.iolib.summary2.Summary'>
"""
                          Results: Logit
=================================================================
Model:              Logit            No. Iterations:   6.0000
Dependent Variable: converted        Pseudo R-squared: 0.000
Date:               2022-06-18 19:21 AIC:              212782.9124
No. Observations:   290585           BIC:              212846.3903
Df Model:           5                Log-Likelihood:   -1.0639e+05
Df Residuals:       290579           LL-Null:          -1.0639e+05
Converged:          1.0000           Scale:            1.0000
-----------------------------------------------------------------
              Coef.   Std.Err.      z      P>|z|    [0.025   0.975]
-----------------------------------------------------------------
intercept    -1.9922    0.0161  -123.4571  0.0000  -2.0238  -1.9606
ab_page       0.0108    0.0228     0.4749  0.6349  -0.0339   0.0555
UK           -0.0118    0.0398    -0.2957  0.7674  -0.0899   0.0663
CA            0.0057    0.0188     0.3057  0.7598  -0.0311   0.0426
ab_UK        -0.0783    0.0568    -1.3783  0.1681  -0.1896   0.0330
ab_CA        -0.0314    0.0266    -1.1811  0.2375  -0.0835   0.0207
=================================================================

"""
```

As p_value for both UK and CA is greater than 0.05 then in another way we can not say that the conversion rate of new page is higher than the old page.

1.0.8 The summary results

I thought that the interaction between ap_page and the country will affect the conversion rate even in any country but actually it didn't , so again we don't have evedience that the new page lead more conversion rate.

Final Check!

Congratulations! You have reached the end of the A/B Test Results project! You should be very proud of all you have accomplished!

Submission You may either submit your notebook through the "SUBMIT PROJECT" button at the bottom of this workspace, or you may work from your local machine and submit on the last page of this project lesson.

1. 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).

2. 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.

3. 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 [50]: from subprocess import call
         call(['python', '-m', 'nbconvert', 'Analyze_ab_test_results_notebook.ipynb'])
```

```
Out[50]: 0
```

```
In [ ]:
```