Analyze_ab_test_results_notebook

March 22, 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 [4]: 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.

Tip: Please save your work regularly.

a. Read in the dataset from the `ab_data.csv` file and take a look at the top few rows here:

```
In [5]: df=pd.read_csv('ab_data.csv')
        df.head()
```

Out[5]:

	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 [6]: df.shape[0]
```

```
Out[6]: 294478
```

c. The number of unique users in the dataset.

```
In [7]: df['user_id'].nunique()
```

```
Out[7]: 290584
```

d. The proportion of users converted.

```
In [8]: df['converted'].mean()
```

```
Out[8]: 0.11965919355605512
```

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

```
In [9]: treatment_not_new =df[(df['group']=='treatment')&(df['landing_page']!= 'new_page') ].cou
        treatment_not_new
```

```
Out[9]: user_id         1965
        timestamp       1965
        group           1965
        landing_page    1965
        converted       1965
        dtype: int64
```

```
In [10]: control_not_old=df[(df['group']=='control')&(df['landing_page']!= 'old_page') ].count()
         control_not_old
```

```
Out[10]: user_id         1928
         timestamp       1928
         group           1928
         landing_page    1928
         converted       1928
         dtype: int64
```

```
In [11]:  # no.of user when in which treatment doesn't match new consist of two part

          treatment_not_match_new=treatment_not_new + control_not_old
          treatment_not_match_new

Out[11]: user_id         3893
         timestamp       3893
         group           3893
         landing_page    3893
         converted       3893
         dtype: int64
```

f. Do any of the rows have missing values?

```
In [12]: df.info() # no missing value

<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


In [13]: df['group'].value_counts()       #no missing value

Out[13]: treatment    147276
         control      147202
         Name: group, dtype: int64

In [14]: df['landing_page'].value_counts()                #no missing value

Out[14]: new_page    147239
         old_page    147239
         Name: landing_page, dtype: int64
```

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 [15]: # Remove the inaccurate rows, and store the result in a new dataframe df2
         df2=df[((df['group']=='treatment')&(df['landing_page']== 'new_page'))|((df['group']=='c
         df2.shape

Out[15]: (290585, 5)

In [16]: # 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].sh

Out[16]: 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 [17]: df2['user_id'].nunique()

Out[17]: 290584
```

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

```
In [18]: df2['user_id'].mode()[0]

Out[18]: 773192
```

c. Display the rows for the duplicate **user_id**?

```
In [19]: df_duplicate_user =df2[df2['user_id']==df2['user_id'].mode()[0]]
         df_duplicate_user

Out[19]:       user_id                   timestamp      group landing_page  converted
         1899   773192  2017-01-09 05:37:58.781806  treatment     new_page          0
         2893   773192  2017-01-14 02:55:59.590927  treatment     new_page          0
```

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

```
In [20]: # 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


         df2=df2[df2['timestamp']!=df_duplicate_user['timestamp'].max()]
```

```
In [21]: df2.shape

Out[21]: (290584, 5)

In [22]: # Check again if the row with a duplicate user_id is deleted or not
         df2[df2['user_id']==df2['user_id'].mode()[0]]

Out[22]:        user_id                   timestamp      group landing_page  converted
         63114   630000  2017-01-19 06:26:06.548941  treatment     new_page          0
```

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?

> **Tip**: The probability you'll compute represents the overall "converted" success rate in the population and you may call it $p_{population}$.

```
In [23]: df2.head()

Out[23]:    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]: p_population =df2.converted.mean()
         p_population

Out[24]: 0.11959708724499628
```

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

```
In [25]: p_control = df2[df2['group']=='control'].converted.mean()
         p_control

Out[25]: 0.1203863045004612
```

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

```
In [26]: p_treatment = df2[df2['group']=='treatment'].converted.mean()
         p_treatment

Out[26]: 0.11880806551510564
```

> **Tip**: The probabilities you've computed in the points (b). and (c). above can also be treated as conversion rate. Calculate the actual difference (`obs_diff`) between the conversion rates for the two groups. You will need that later.

```
In [27]: # Calculate the actual difference (obs_diff) between the conversion rates for the two g
         obs_diff = p_treatment - p_control
         obs_diff

Out[27]: -0.0015782389853555567
```

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

```
In [28]: df2[df2['landing_page']=='new_page'].count()/df2.count()

Out[28]: 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 the new `treatment` group users lead to more conversions.

> **It's obvious from the difference between the conversion rate of treatment group and control group that there is no clear difference or a slight difference in favor of control group, so this result does not support the new page at this level of the study.**

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.

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.

H_0 : p_{new} - p_{old} **0**

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

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 [29]: p_population=df2.converted.mean()
         p_population

Out[29]: 0.11959708724499628

In [30]: p_new = p_population   # as   =    =
         p_new

Out[30]: 0.11959708724499628
```

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

```
In [31]: p_old = p_population      # as   =    =
         p_old

Out[31]: 0.11959708724499628
```

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

```
In [32]: df_treatment = df2[df2['group']=='treatment']
         n_new=df_treatment.shape[0]
         n_new

Out[32]: 145310
```

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

```
In [33]: df_control =df2[df2['group']=='control']
         n_old = df_control.shape[0]
         n_old
```

```
Out[33]: 145274
```

e. Simulate Sample for the `treatment` Group Simulate n_{new} transactions with a conversion rate of p_{new} under the null hypothesis. *Hint*: Use `numpy.random.choice()` method to randomly generate n_{new} number of values. Store these n_{new} 1's and 0's in the `new_page_converted` numpy array.

```
In [34]: # Simulate a Sample for the treatment Group
         new_page_converted=df_treatment.converted
         treatment_sample = np.random.choice(new_page_converted,n_new,replace=True)
         treatment_sample

Out[34]: array([0, 0, 0, ..., 1, 0, 0])
```

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 [35]: # Simulate a Sample for the control Group

         old_page_converted=df_control.converted
         control_sample =np.random.choice(old_page_converted,n_old,replace=True)
         control_sample

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

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

```
In [36]: diff_probability=treatment_sample.mean()-control_sample.mean()
         diff_probability

Out[36]: 0.00019071079244332989
```

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 [96]: # Sampling distribution
         df_new = df2[df2['group']=='treatment']
         n_new=df_new.shape[0]


         df_old =df2[df2['group']=='control']
         n_old = df_old.shape[0]



         p_diffs = []
```

```
for i in range(10000):
    sample_new =np.random.choice(df_new['converted'],size=n_new,replace=True)
    sample_old =np.random.choice(df_old['converted'],size=n_old,replace=True)
    p_diffs.append(sample_new.mean()-sample_old.mean())


p_diffs=np.array(p_diffs)
std = p_diffs.std()
```

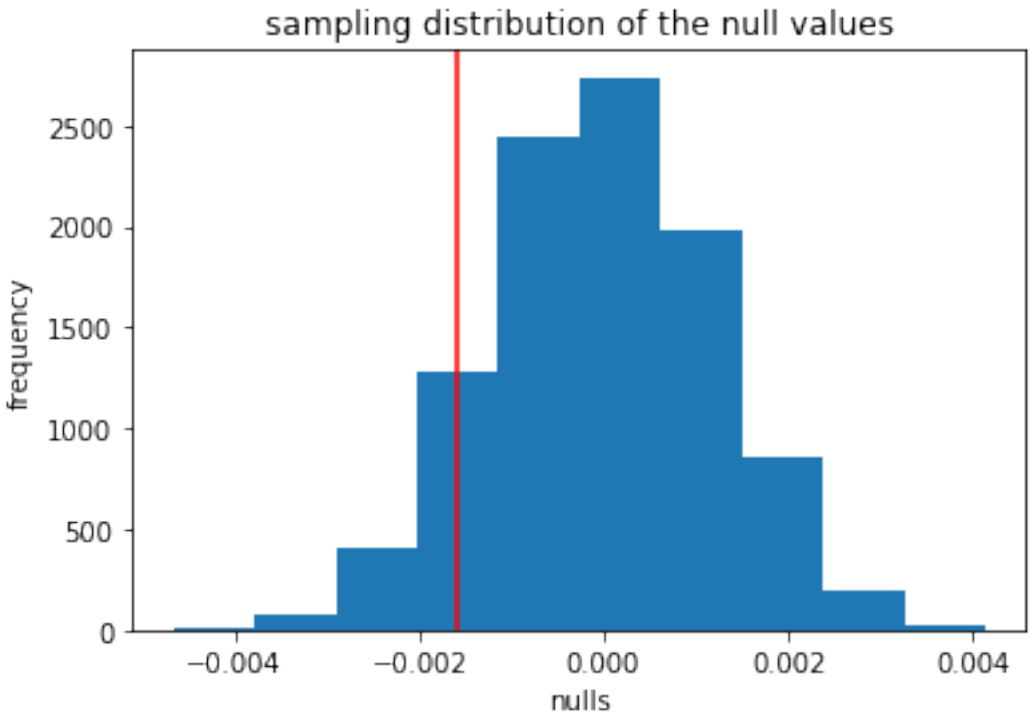
i. Histogram Plot a histogram of the **p_diffs**. Does this plot look like what you expected? Use the matching problem in the classroom to assure you fully understand what was computed here.

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

In [97]:
```
#distribut of the null values
nulls=np.random.normal(0,std,p_diffs.size)
```

In [98]:
```
plt.hist(nulls)
plt.axvline(obs_diff,color = 'red')
plt.title('sampling distribution of the null values')
plt.xlabel('nulls')
plt.ylabel('frequency')
```

Out[98]:
```
Text(0,0.5,'frequency')
```



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

```
In [99]: (nulls>obs_diff).mean()
```

```
Out[99]: 0.9032
```

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)".

> ** the value appear here is the p_value which tell us the proportion of the values of p_diffs that occure due to chance not due to a real difference , so this large p_value far more than 0.05 (type 1 error) indicate that no significant difference and we can not reject the null hypothesis**

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 [40]: import statsmodels.api as sm

         # number of conversions with the old_page
         convert_old = df2[df2['landing_page']=='old_page']['converted'].sum()

         # number of conversions with the new_page
         convert_new =df2[df2['landing_page']=='new_page']['converted'].sum()

         # number of individuals who were shown the old_page
         n_old = df2[df2['landing_page']=='old_page']['converted'].count()

         # number of individuals who received new_page
         n_new = df2[df2['landing_page']=='new_page']['converted'].count()

         convert_old , convert_new, n_old ,n_new
```

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

```
Out[40]: (17489, 17264, 145274, 145310)
```

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.

> **Tip**: You don't have to dive deeper into z-test for this exercise. **Try having an overview of what does z-score signify in general.**

```
In [41]: 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],a
         print(z_score, p_value)
```

```
-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.**?

> **Tip**: Notice whether the p-value is similar to the one computed earlier. Accordingly, can you reject/fail to reject the null hypothesis? It is important to correctly interpret the test statistic and p-value.

> ** this study is a right tailed as our altarnative hypothesis idicate that the difference between the conersion rate > 0 so our critical value for 95% confidence interval where Z0.05 is 1.96 standard error while the z score here appear to be -1.31 so it is not in the rejection region also the p value > 0.05 so it is a high value indicate not rejecting null hypothesis

**

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?

> **since it is two category converted and not converted (Y/N) so it follows logestic 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 [42]: df2.head()
```

```
Out[42]:    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 [43]: df2[['ab_page','old_page']]=pd.get_dummies(df2['landing_page'])
         df2=df2.drop('old_page',axis=1)

In [44]: df2.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
         0        0
         1        0
         2        1
         3        1
         4        0
```

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 [45]: df2['intercept']=1
         log_model=sm.Logit(df2['converted'],df2[['intercept','ab_page']])
         result=log_model.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 [46]: result.summary2()

Out[46]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                 Results: Logit
         =================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
```

```
Date:               2022-02-27 08:18 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**?

as clear here the p_value associated with ab_page is 0.1899 which is large enough to to prove no significant effect for changing the page and we notice here that this p_value differ from the p_value we get from the previous test and this is due to diferent hypothisis in these tests while the alternative hypothesis in the first test was that the difference in conversion rate more than 0 (one tailled) here the altarnative hypothesis is that the slope that describe the relation between the conversion rate and the page not equal to 0 (two tailed) so there is difference in the hypothesis and in the measuring tool we use to measure.

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?

from our study we don't find an effect for different pages on th conversion rate so it is a good idea to search and try another factor and test it to discover if any other factor affect the result but we have to take care of some disadvantage that may raise like non linearity , multicolinearity ,outliers and correlated errors.

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. >**Hint:** Use `pandas.get_dummies()` to create dummy variables. **You will utilize two columns for the three dummy variables.**

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

```
In [47]: # Read the countries.csv
         df3=pd.read_csv('countries.csv')

In [48]: # Join with the df2 dataframe
         df_merged= df2.merge(df3,on='user_id')

In [49]: # Create the necessary dummy variables
         df_merged[['CA', 'UK', 'US']]=pd.get_dummies(df_merged['country'])
         df_merged.head()
```

Out[49]:

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

```
In [52]: # Fit your model, and summarize the results
         df_merged['intercept']=1 #it is already available
         log_model = sm.Logit(df_merged['converted'],df_merged[['intercept','ab_page','CA','UK']
         result=log_model.fit()
         result.summary2()

Optimization terminated successfully.
         Current function value: 0.366113
         Iterations 6


Out[52]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                  Results: Logit
         =================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2022-02-27 08:18 AIC:              212781.1253
         No. Observations:   290584           BIC:              212823.4439
         Df Model:           3                Log-Likelihood:   -1.0639e+05
         Df Residuals:       290580           LL-Null:          -1.0639e+05
         Converged:          1.0000           Scale:            1.0000
         -----------------------------------------------------------------
                       Coef.   Std.Err.      z      P>|z|    [0.025   0.975]
         -----------------------------------------------------------------
```

```
intercept    -1.9893    0.0089  -223.7628  0.0000  -2.0067  -1.9718
ab_page      -0.0149    0.0114    -1.3069  0.1912  -0.0374   0.0075
CA           -0.0408    0.0269    -1.5161  0.1295  -0.0934   0.0119
UK            0.0099    0.0133     0.7433  0.4573  -0.0162   0.0359
=================================================================

"""
```

As clear from the P_value for each country that no statistically significant correlation between country and conversion rate, so we will try to get linearity though adding interaction on our model

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.

> **Tip**: Conclusions should include both statistical reasoning, and practical reasoning for the situation.

```
In [54]: df_merged['ab_CA']=df_merged['ab_page']*df_merged['CA']
```

```
In [57]: df_merged['ab_UK']=df_merged['ab_page']*df_merged['UK']
```

```
In [63]: df_merged['intercept']=1 #it is already available
         #while we include higher order , we have to include also lower order
         log_model = sm.Logit(df_merged['converted'],df_merged[['intercept','ab_page','CA','UK',
         result=log_model.fit()
         result.summary2()
```

```
Optimization terminated successfully.
         Current function value: 0.366109
         Iterations 6
```

```
Out[63]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                  Results: Logit
         =================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2022-02-27 09:06 AIC:              212782.6602
         No. Observations:   290584           BIC:              212846.1381
         Df Model:           5                Log-Likelihood:   -1.0639e+05
         Df Residuals:       290578           LL-Null:          -1.0639e+05
         Converged:          1.0000           Scale:            1.0000
         -----------------------------------------------------------------
                      Coef.   Std.Err.      z      P>|z|    [0.025   0.975]
```

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

"""
```

```
In [51]: df_merged['timestamp'].min(),df_merged['timestamp'].max() # the range of time the exper
```

```
Out[51]: ('2017-01-02 13:42:05.378582', '2017-01-24 13:41:54.460509')
```

** after including the country in our model and the interaction , we don't find any statisticaly significant result that support the alternative hypothesis as all p_value > 0.05 so we can't reject the null hypothesis that the slop = 0 which mean that ther is no relation between the different country or the different pages on the conversion rate, practically we see that the study only done for 22 day that may include other effect like seasonal effect , may be if the study occure in different time or on larger time scale , may get different result.**

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

```
Out[100]: 0
```