Analyze_ab_test_results_notebook

December 12, 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 necessary libraries
        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.nunique()['user_id']
```

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

d. The proportion of users converted.

```
In [5]: df['converted'].sum()/df.nunique()['user_id']
```

```
Out[5]: 0.12126269856564711
```

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

```
In [6]: group_t=df.query('group=="treatment" and landing_page!="new_page"')['user_id'].count()
        group_t
```

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

f. Do any of the rows have missing values?

```
In [7]: 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
```

from the above resulst there are no missing values

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]: # Remove the inaccurate rows, and store the result in a new dataframe df2
        # we will use the append function to get the required matching
        df2=df.query("group=='control'and landing_page!='new_page'")
        df2=df2.append(df.query("group=='treatment'and landing_page=='new_page'"))
        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
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
7	719014	2017-01-17 01:48:29.539573	control	old_page	0

```
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
```

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 [10]: df2.nunique()['user_id']

Out[10]: 290584

In [11]: 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
```

no missing values.

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

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

Out[12]:

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

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

```
In [13]: df2.get(df2['user_id'] == 773192)
```

Out[13]:

	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 [14]: # we will Remove one of the rows with a duplicate user_id.
         df2=df2.drop(index=1899, axis=0)
         # we will Check again if the row with a duplicate user_id is deleted or not
         df2.get(df2['user_id'] == 773192)
```

Out[14]:

	user_id	timestamp	group	landing_page	converted
2893	773192	2017-01-14 02:55:59.590927	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?

```
In [15]: df2['converted'].mean()
```

```
Out[15]: 0.11959708724499628
```

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

```
In [16]: control_P=df2.query('group=="control"')['converted'].mean()
         control_P
```

```
Out[16]: 0.1203863045004612
```

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

```
In [17]: tret_P=df2.query('group=="treatment"')['converted'].mean()
         tret_P

Out[17]: 0.11880806551510564
```

Calculate the actual difference (`obs_diff`) between the conversion rates for the two groups. You will need that later.

```
In [18]: # we Calculate the actual difference (obs_diff) between the conversion rates for the tu
         obs_diff_P=tret_P - control_P
         obs_diff_P

Out[18]: -0.0015782389853555567
```

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

```
In [19]: total_received=df2.shape[0]
         received_new =df2.query('landing_page=="new_page"').shape[0]/total_received
         received_new

Out[19]: 0.5000619442226688
```

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

the new treatment group did not lead to more convertion rate than the control group as the obs_diff difference between them is postive (0.001578) despite it is very low and may considered negligible.

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.

NULL H0: P >= Pnew

ALTERNATIVE H1: Pold < Pnew.

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 [20]: p_population =df2['converted'].mean()
         p_new =p_population # as   =  =  under the null
         p_new

Out[20]: 0.11959708724499628
```

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

```
In [21]: p_old =p_population # as   =  =  under the null
         p_old

Out[21]: 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 [22]: n_new=df2.query('group=="treatment"').count()['user_id']
         n_new

Out[22]: 145310
```

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

```
In [23]: n_old=df2.query('group=="control"').count()['user_id']
         n_old

Out[23]: 145274

In [24]: total=n_new+n_old
         total
```

```
Out[24]: 290584
```

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

```
In [25]: # we will Simulate a Sample for the treatment Group
         new_page_converted=np.random.choice([1,0], n_new, replace=True , p=[p_new,(1-p_new)])
         sim_new_p=new_page_converted.mean()
         sim_new_p
```

```
Out[25]: 0.1196751772073498
```

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 [26]: # we will Simulate a Sample for the control Group
         old_page_converted=np.random.choice([1,0], n_old, replace=True, p=[p_old, (1-p_old)])
         sim_old_p=old_page_converted.mean()
         sim_old_p
```

```
Out[26]: 0.11886504123242975
```

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

```
In [27]: sim_new_p - sim_old_p
```

```
Out[27]: 0.00081013597492005096
```

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.

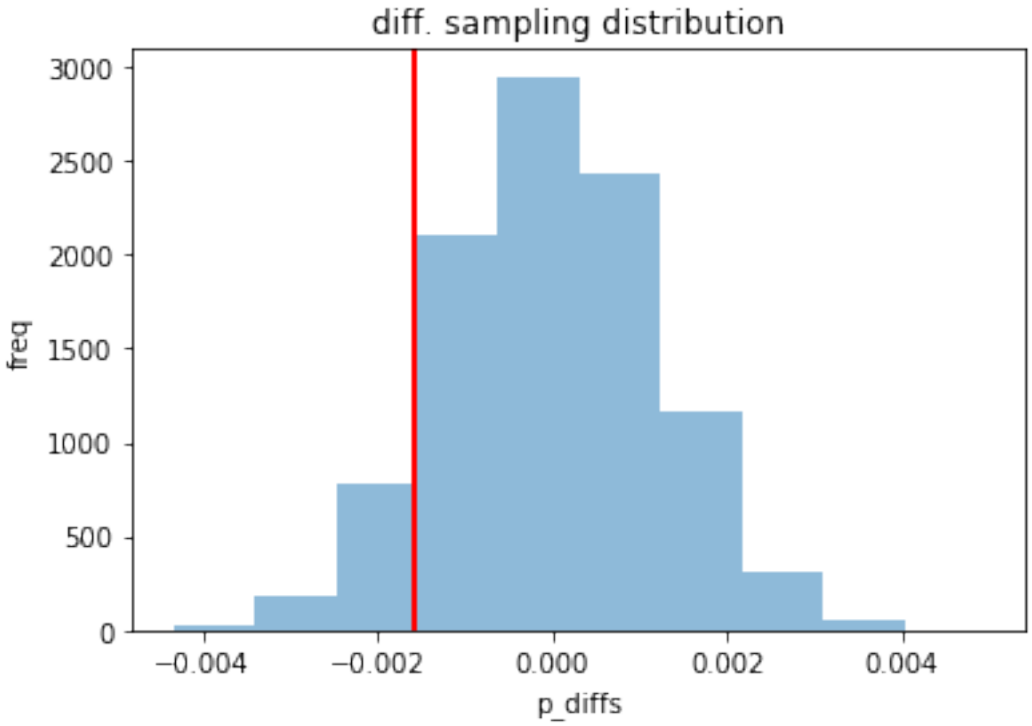
Store all $(p'_{new}$ - $p'_{old})$ values in a NumPy array called `p_diffs`.

```
In [28]: # Sampling distribution
         p_diffs = []
         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
         p_diffs = new_converted_simulation - old_converted_simulation
```

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 [29]: plt.hist(p_diffs, alpha=0.5);
         plt.xlabel('p_diffs');
         plt.ylabel('freq');
         plt.title('diff. sampling distribution');
         plt.axvline(obs_diff_P, color='r', linewidth=2);
```



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

```
In [30]: (p_diffs > obs_diff_P).mean()
```

```
Out[30]: 0.90480000000000005
```

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)".

first we calcualte the observed difference in our sample population(obs_diff_P) then we get a random sampling distribution from our sample for the difference in proportin of convertion (p_diffs) to calculate the p_value (which is the probability of observing our statistisc or extremes if null hypothesis is true) hence the p_value (0.9) is larger than type 1 error rate (0.05) we fail to reject the null hypothesis and it is more likely that our statistic come form the null not from the alternative 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 [31]: import statsmodels.api as sm
```

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

        # number of conversions with the new_page
        convert_new = df2.query('landing_page=="new_page"')['converted'].sum()
        convert_old , convert_new
```

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

```
Out[31]: (17489, 17264)
```

```
In [32]: # number of individuals who were shown the old_page
         n_old = df2.query('landing_page=="old_page"').count()['user_id']

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

```
Out[32]: (145274, 145310)
```

```
In [33]: #total converted for each group
         total_convert=np.array([convert_old,convert_new])
         total_convert
```

```
Out[33]: array([17489, 17264])
```

```
In [34]: #total observation for each group
         total_observation=np.array([n_old,n_new])
         total_observation
```

```
Out[34]: array([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.

```
In [35]: import statsmodels.api as sm
         # we will 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)
```

```
-1.31092419842 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.**?

z-score is used to compare the distance between our statistic (difference between convertion rate of the two groups) from the null according to the standard error. here is lower than (<) so fail to reject the null hypothesis. also the calaculated p-value here(0.9) is matching with that calculated earlier (0.9) which is also still higher than the type 1 error rate (0.05) so we fail to reject the null hypothesis. at the end the calculated and p.value are matching with the prevoius findings.

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?

logistic regression as we predict a categorical response with two possible outcomes converted or not converted.

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 [36]: #create dummy varaibles
         df2[['control', 'ab_page']]=pd.get_dummies(df2['group'])
         df2=df2.drop('control', axis=1)
         df2.head()
```

```
Out[36]:    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
         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
         7   719014  2017-01-17 01:48:29.539573  control     old_page          0

            ab_page
         0        0
         1        0
         4        0
         5        0
         7        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 [37]: #create logistic regression model
         df2['intercept']=1
         logit_mod=sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])

         #fit the model
         results=logit_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 [38]: #get summary statistic
         results.summary2()

Out[38]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                 Results: Logit
         =================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2022-12-12 01:07 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**?

Here p-value of ab_page = 0.1899 and it looks little different than that we calculated before this may due to change in the setting of null and alternative hypothesis associated with logistic regression (here in logistic regression we assume that the converion is depend on ab_page) while in the prevoius section for the null and alternative hypothesises we assume there is an equality in conversion regardless the page view. also the hypothesis in part1 is one-sided while in part3 is two-sided. The current p-value is still higher than type 1 error rate (0.05) like before we fail to reject the null hypothesis.

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?

yes i see it is agood idea to consider other factors (as factors here have little impact on conversion prediction) to be add to our logistic regression as this may reveal an hidden factors that may contribute to more impact on conversion prediction. we should take care when adding other factors, we want them not to be correlated with each others to avoid multi-collinearity as this lead to unrealibale analysis.

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 [39]: # Read the countries.csv
         new_df=pd.read_csv('countries.csv')
         new_df.head()
```

```
Out[39]:    user_id country
         0   834778      UK
         1   928468      US
         2   822059      UK
         3   711597      UK
         4   710616      UK
```

```
In [40]: # Join with the df2 dataframe
         df_merged=df2.join(new_df.set_index('user_id'), on=('user_id'))
         df_merged.head()
```

```
Out[40]:    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
         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
         7   719014  2017-01-17 01:48:29.539573  control     old_page          0

            ab_page  intercept country
         0        0          1      US
         1        0          1      US
         4        0          1      US
         5        0          1      US
         7        0          1      US
```

```
In [41]: # Create the necessary dummy variables
         df_merged[['UK', 'US', 'CA']]=pd.get_dummies(df_merged['country'])
         df_merged=df_merged.drop('CA', axis=1)
         df_merged.head()
```

```
Out[41]:    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
         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
         7   719014  2017-01-17 01:48:29.539573  control     old_page          0

            ab_page  intercept country  UK  US
         0        0          1      US   0   0
         1        0          1      US   0   0
         4        0          1      US   0   0
         5        0          1      US   0   0
         7        0          1      US   0   0
```

```
In [42]: # Fit the model, and summarize the results
         logit_mod=sm.Logit(df_merged['converted'],df_merged[['intercept','ab_page', 'UK','US']])
         results=logit_mod.fit()
         results.summary2()

Optimization terminated successfully.
         Current function value: 0.366113
         Iterations 6


Out[42]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                  Results: Logit
         =================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2022-12-12 01:07 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
         UK           -0.0408    0.0269    -1.5161  0.1295  -0.0934   0.0119
         US            0.0099    0.0133     0.7433  0.4573  -0.0162   0.0359
         =================================================================

         """
```

by looking at the resulted p-values of the countries (they are higher than type 1 error rate 0.05) so we fail to reject the null hypothesis.

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 [43]: # here we will see the effect of page and country interaction on conversion prediction
         df_merged['UK_ab_page']=df_merged ['UK']*df_merged['ab_page']
         df_merged['US_ab_page']=df_merged ['US']*df_merged['ab_page']

In [44]: #fit the model
         logit_mod2=sm.Logit(df_merged['converted'],df_merged[['intercept','ab_page','UK','US','
         results2=logit_mod2.fit()
         results2.summary2()

Optimization terminated successfully.
         Current function value: 0.366109
         Iterations 6


Out[44]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                  Results: Logit
         =================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2022-12-12 01:07 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
         UK           -0.0175    0.0377    -0.4652  0.6418  -0.0914   0.0563
         US           -0.0057    0.0188    -0.3057  0.7598  -0.0426   0.0311
         UK_ab_page   -0.0469    0.0538    -0.8718  0.3833  -0.1523   0.0585
         US_ab_page    0.0314    0.0266     1.1807  0.2377  -0.0207   0.0835
         =================================================================

         """
```

here we can see that all the calculated p-values are higher than type 1 error rate (0.05) so there is no statistically significant effect for the country and page interaction on conversion prediction so we fail to reject the null hypothesis.

finally we can conclude from the above calculations there is no need to switch from old page to new page as the new page fail to drive new users conversion. the old page is good to keep for now until circumstances change that drive the need for new experiment

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 [45]:
```
from subprocess import call
call(['python', '-m', 'nbconvert', 'Analyze_ab_test_results_notebook.ipynb'])
```

Out[45]:
```
0
```