AB_results

May 4, 2021

0.1 Project description

This project have mastered the subjects covered in the statistics lessons.The hope is to have this project be as comprehensive of these topics as possible.

0.2 Table of Contents

0.3 Project purpose

We will be working to understand the results of an A/B test run by an e-commerce website. Our goal is to work through this notebook to help the company understand if they should implement the new page, keep the old page, or perhaps run the experiment longer to make their decision.

[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)
```

0.4 Part I- Probability

1.A. Read in the dataset and take a look at the top few rows here:

[2]:
```
user=pd.read_csv('ab_data.csv')
display(user.head())
```

[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

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

[3]:
```
print('number of rows:',user.shape[0])
```

```
number of rows: 294478
```

1.C. The number of unique users in the dataset.

[4]:
```
print('number of unique users:',user['user_id'].nunique())
```

```
number of unique users: 290584
```

1.D. The proportion of users converted.

[5]:
```
proportion = (user.query('converted ==1')['user_id'].nunique())/
 ↪(user['user_id'].nunique())
print(proportion)
```

```
0.12104245244060237
```

1.E. The number of times the new_page and treatment don't line up.

[6]:
```
mismatch= user.query('(group== "treatment") != (landing_page== "new_page")')
print('number of times the new_page and treatment do not match:',mismatch.
 ↪shape[0])
```

```
number of times the new_page and treatment do not match: 3893
```

1.F. Do any of the rows have missing values?

[7]:
```
display(user.isnull().sum())
```

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

2. Now use the answer to the quiz to create a new dataset that meets the specifications from the quiz. Store your new dataframe in user_2.

[8]:
```
user_2= user.query('((group=="control") & (landing_page=="old_page")) | \
                    (group=="treatment") & (landing_page=="new_page") ')
print(user_2.shape[0])
```

```
290585
```

3.a. How many unique user_ids are in user_2?

[9]:
```
print('number of unique users:',user_2['user_id'].nunique())
```

```
number of unique users: 290584
```

3.b. There is one user_id repeated in user_2. What is it?

```
[10]: user_2['is_duplicated'] = user_2.duplicated(['user_id'])
      user_2['is_duplicated'].value_counts()
```

```
<ipython-input-10-842195aa1926>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  user_2['is_duplicated'] = user_2.duplicated(['user_id'])
```

```
[10]: False    290584
      True          1
      Name: is_duplicated, dtype: int64
```

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

```
[11]: user_2_dup = user_2.loc[user_2['is_duplicated'] == True]
      display(user_2_dup)
```

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

            is_duplicated
      2893           True
```

3.d. Remove one of the rows with a duplicate user_id, but keep your dataframe as user_2.

```
[12]: user_2.drop_duplicates("user_id", inplace=True)
      user_2.head()
```

```
<ipython-input-12-d43f0a235b61>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  user_2.drop_duplicates("user_id", inplace=True)
```

```
[12]:    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
```

```
   is_duplicated
0          False
1          False
2          False
3          False
4          False
```

4.a. What is the probability of an individual converting regardless of the page they receive?

[13]:
```
# since values are 1 and 0, we can calculate mean to get probability of an␣
↪individual converting
individual_probabilty= user_2['converted'].mean()
print('individual_probabilty:',individual_probabilty)
```

```
individual_probabilty: 0.11959708724499628
```

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

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

[14]:
```
user_2_grp = user_2.groupby('group')
display(user_2_grp.describe())
```

[14]:
```
             user_id                                                     \
               count           mean           std       min        25%
group
control     145274.0  788164.072594  91287.914601  630002.0  709279.50
treatment   145310.0  787845.719290  91161.564429  630000.0  708745.75

                                              converted                      \
                50%        75%       max      count      mean       std  min
group
control    788128.5  867208.25  945998.0   145274.0  0.120386  0.325414  0.0
treatment  787876.0  866718.75  945999.0   145310.0  0.118808  0.323564  0.0


           25%  50%  75%  max
group
control    0.0  0.0  0.0  1.0
treatment  0.0  0.0  0.0  1.0
```

1. Given that an individual was in the control group, the probability they converted is 0.120399
2. Given that an individual was in the treatment group, the probability they converted is 0.118920

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

[15]:
```
print((user_2['landing_page'].value_counts())/(user_2.shape[0]))
```

```
new_page    0.500062
old_page    0.499938
Name: landing_page, dtype: float64
```

4.e. Consider your results from a. through d. above, and explain below whether you think there is sufficient evidence to say that the new treatment page leads to more conversions.

No, the treatment group has a less probability than the control group. Therefore, there is no evidence to conclude that the new treatment page leads to more conversions.

0.4.1 Part II - A/B Test

1. For now, consider we need to make the decision just based on all the data provided. If we 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 our null and alternative hypotheses be? We can state your hypothesis in terms of words or in terms of and , which are the converted rates for the old and new pages.

- **Hypothesis**

$$H_0 : p_{new} \leq p_{old}$$

$$H_1 : p_{new} > p_{old}$$

2. Assume under the null hypothesis, and both have "true" success rates equal to the converted success rate regardless of page - that is and are equal. Furthermore, assume they are equal to the converted rate in ab_data.csv regardless of the page.

Use a sample size for each page equal to the ones in ab_data.csv.

Perform the sampling distribution for the difference in converted between the two pages over 10,000 iterations of calculating an estimate from the null.

2. a. What is the conversion rate for p_{new} under the null?

[16]:
```
ab_df=pd.read_csv('ab_data.csv')
display(ab_df.head())

p_new = ab_df['converted'].mean()
print(p_new)
```

[16]:

	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
```

```
0.11965919355605512
```

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

[17]:
```
p_old = ab_df['converted'].mean()
print(p_old)
```

```
0.11965919355605512
```

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

[18]:
```
n_new = len(ab_df.query("group == 'treatment'"))
print(n_new)
```

```
147276
```

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

[19]:
```
n_old = len(ab_df.query("group == 'control'"))
print(n_old)
```

```
147202
```

2.e. Simulate n_{new} transactions with a conversion rate of p_{new} under the null. Store these n_{new} 1's and 0's in new_page_converted.

[20]:
```
ab_df['new_page_converted'] = ab_df.query('landing_page == "new_page"').
 ↪converted
```

2.f. Simulate n_{old} transactions with a conversion rate of p_{old} under the null. Store these n_{old} 1's and 0's in old_page_converted.

[21]:
```
ab_df['old_page_converted'] = ab_df.query('landing_page == "old_page"').
 ↪converted
```

[22]:
```
display(ab_df.head())
```

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

   new_page_converted  old_page_converted
```

0	NaN	0.0
1	NaN	0.0
2	0.0	NaN
3	0.0	NaN
4	NaN	1.0

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

[23]:
```
diff_new = ab_df['new_page_converted'].mean() - ab_df['old_page_converted'].
↪mean()
display(diff_new)
```

[23]: -0.0016367945992569882

2.h. Simulate 10,000 p_{new} - p_{old} values using this same process similarly to the one you calculated in parts a. through g. above. Store all 10,000 values in a numpy array called p_diffs.

[24]:
```
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
```

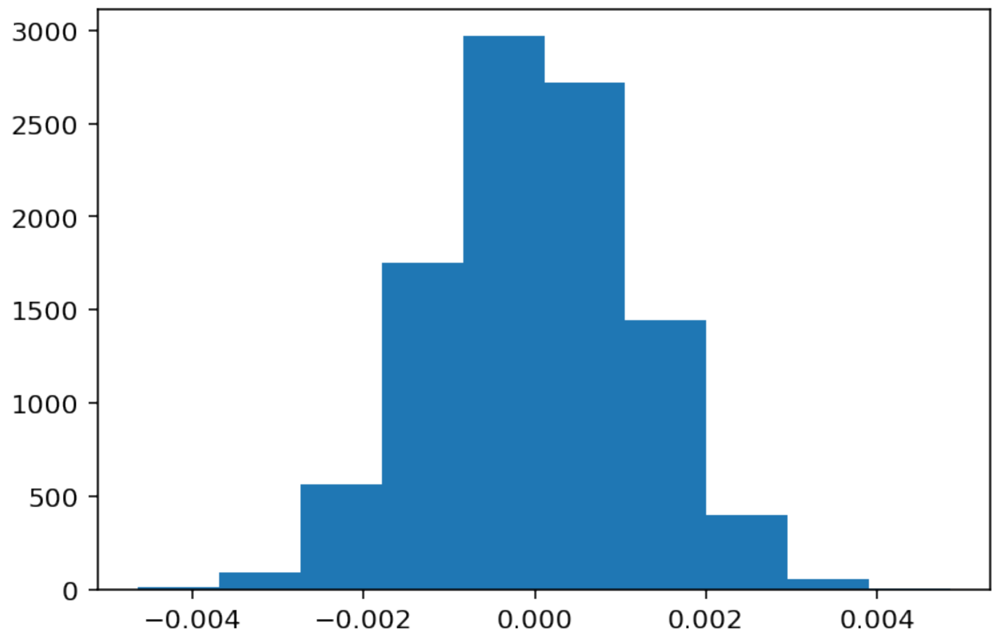
[25]:
```
p_diffs = np.array(p_diffs)
```

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

[26]:
```
plt.hist(p_diffs)
```

[26]:
```
(array([  12.,   91.,  560., 1751., 2967., 2716., 1443.,  396.,   57.,
           7.]),
 array([-0.00463071, -0.00368261, -0.00273452, -0.00178642, -0.00083832,
         0.00010977,  0.00105787,  0.00200597,  0.00295406,  0.00390216,
         0.00485026]),
 &lt;BarContainer object of 10 artists&gt;)
```

[26]:



This graph follows the normal distribution. It is because of the central limit theorem

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

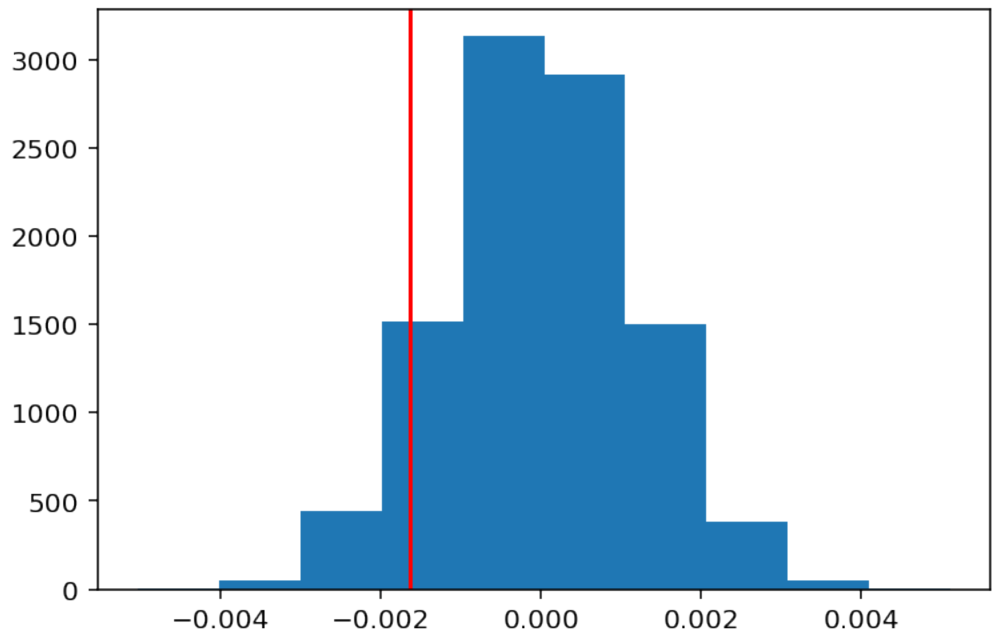
[27]:
```
(p_diffs > diff_new).mean()
```

[27]: 0.9164

[28]:
```
null_mean = 0
null_vals = np.random.normal(null_mean, p_diffs.std(), 10000)
plt.hist(null_vals);

plt.axvline(x=diff_new, color = 'red');
```

[28]:



[29]:
```
p_val = (null_vals > diff_new).mean()
p_val
```

[29]: 0.9128

2.k. In words, explain what you just computed in part j.. What is this value called in scientific studies? What does this value mean in terms of whether or not there is a difference between the new and old pages?

1. The above right line is where our observed statistics fall, the value I just computed in part j is the p-value.
2. This p-value is greater than 0.05 so that we cannot reject the null hypothesis. We can conclude there is not differene between the new and old pages

2.l. We could also use a built-in to achieve similar results. Though using the built-in might be easier to code, the above portions are a walkthrough of the ideas that are critical to correctly thinking about statistical significance. Fill in the below to calculate the number of conversions for each page, as well as the number of individuals who received each page. Let n_old and n_new refer the the number of rows associated with the old page and new pages, respectively.

[30]:
```
import statsmodels.api as sm

convert_old = ab_df.query('landing_page == "old_page"').converted.sum()
convert_new = ab_df.query('landing_page == "new_page"').converted.sum()
n_old = ab_df.query('landing_page == "old_page"').user_id.count()
```

```python
n_new = ab_df.query('landing_page == "new_page"').user_id.count()
print(convert_old)
print(convert_new)
print(n_old)
print(n_new)
```

```
17739
17498
147239
147239
```

[31]:
```python
z_score, p_value = sm.stats.proportions_ztest([convert_old, convert_new],
 ↪[n_old, n_new], alternative='smaller')
print(z_score, p_value)
```

```
1.3683341399998907 0.9143962454534289
```

2.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.?

1. The z-score and p-value communicate the same message as part j and k, our p-value is very large which suggest our statistic is likely to come from the null hypothesis.
2. Hence, we fail to reject the null hypothesis and conclude that new page is not better than old page.

0.4.2 Part III - A regression approach

1.a In this final part, we will see that the result we achieved in the A/B test in Part II above can also be achieved by performing regression.Since each row is either a conversion or no conversion, what type of regression should you be performing in this case? Logistic regression

1.b. The goal is to use statsmodels to fit the regression model we specified in part a. to see if there is a significant difference in conversion based on which page a customer receives. However, we first need to create a column for the intercept, and create a dummy variable column for which page each user received. Add an intercept column, as well as an ab_page column, which is 1 when an individual receives the treatment and 0 if control.

[32]:
```python
display(ab_df.head())
```

[32]:

	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	

```
   new_page_converted  old_page_converted
0                 NaN                 0.0
1                 NaN                 0.0
2                 0.0                 NaN
3                 0.0                 NaN
4                 NaN                 1.0
```

[33]:
```
ab_df['intercept'] = 1
ab_df[['ab_page', 'ab_page_temp']] = pd.get_dummies(ab_df.landing_page)
ab_df.head()
```

[33]:
```
   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

   new_page_converted  old_page_converted  intercept  ab_page  ab_page_temp
0                 NaN                 0.0          1        0             1
1                 NaN                 0.0          1        0             1
2                 0.0                 NaN          1        1             0
3                 0.0                 NaN          1        1             0
4                 NaN                 1.0          1        0             1
```

[34]:
```
ab_df.drop('ab_page_temp', axis=1, inplace=True)
ab_df.head()
```

[34]:
```
   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

   new_page_converted  old_page_converted  intercept  ab_page
0                 NaN                 0.0          1        0
1                 NaN                 0.0          1        0
2                 0.0                 NaN          1        1
3                 0.0                 NaN          1        1
4                 NaN                 1.0          1        0
```

1.c. Use statsmodels to instantiate our regression model on the two columns we created in part b., then fit the model using the two columns we created in part b. to predict whether or not an individual converts.

[35]:
```
import statsmodels.api as sm
logitmod = sm.Logit(ab_df['converted'], ab_df[['intercept', 'ab_page']])
```

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

[36]:
```
results = logitmod.fit()
results.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.366242
         Iterations 6
```

[36]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              converted   No. Observations:               294478
Model:                          Logit   Df Residuals:                   294476
Method:                           MLE   Df Model:                            1
Date:                Tue, 04 May 2021   Pseudo R-squ.:               8.680e-06
Time:                        14:04:40   Log-Likelihood:            -1.0785e+05
converged:                       True   LL-Null:                   -1.0785e+05
Covariance Type:            nonrobust   LLR p-value:                    0.1712
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -1.9879      0.008   -248.305      0.000      -2.004      -1.972
ab_page       -0.0155      0.011     -1.368      0.171      -0.038       0.007
==============================================================================
"""
```

1.e. What is the p-value associated with ab_page? Why does it differ from the value we found in Part II? Hint: What are the null and alternative hypotheses associated with our regression model, and how do they compare to the null and alternative hypotheses in the Part II?

- **Hypothesis**

$$H_0 : p_{new} - p_{old} = 0$$

$$H_1 : p_{new} - p_{old} \neq 0$$

The p-value associated with ab_page is 0.171. This is because the approach of calculating the p-value is different for each case. For the first case we calculate the probability receiving a observed statistic if the null hypothesis is true. Therefore this is a one-sided test. However, the ab_page p-value is the result of a two sided test, because the null hypothesis for this case is, here we are asking whether there is a difference in conversion rate between new page and old page.

Based on that p_value we can say, that the conversion is not significant dependent on the page.

1.f. Now, we 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 our regression model. Are there any disadvantages to adding additional terms into our regression model?

1. It is a good idea to consider other factors to add into our regression model ,for example the day of the week or the gender/income infrastructure (if this data would be available)which could extract from the time stamp. This could lead to more precise results and a higher accuracy.
2. The disadvantages to adding additional terms into the regression model is that even with additional factors we can never account for all influencing factors or accomodate them.
3. Multicolinearity on the other hand is more troublesome to detect because it emerges when three or more variables, which are highly correlated, are included within a model.

1.g. Now along with testing if the conversion rate changes for different pages, also add an effect based on which country a user lives in. You will need to read in the countries.csv dataset and merge together your datasets on the appropriate rows. Here are the docs for joining tables.

Does it appear that country had an impact on conversion? Don't forget to create dummy variables for these country columns - Hint: You will need two columns for the three dummy variables. Provide the statistical output as well as a written response to answer this question.

[37]:
```
ab_df_countries = pd.read_csv("countries.csv")
display(ab_df_countries.head())
```

[37]:

	user_id	country
0	834778	UK
1	928468	US
2	822059	UK
3	711597	UK
4	710616	UK

[38]:
```
#merge the dataframes together
ab_df_log_country = ab_df_countries.merge(ab_df, on="user_id", how = "left")
display(ab_df_log_country.head())
```

[38]:

	user_id	country	timestamp	group	landing_page	\
0	834778	UK	2017-01-14 23:08:43.304998	control	old_page	
1	928468	US	2017-01-23 14:44:16.387854	treatment	new_page	
2	822059	UK	2017-01-16 14:04:14.719771	treatment	new_page	
3	711597	UK	2017-01-22 03:14:24.763511	control	old_page	

```
4   710616      UK  2017-01-16 13:14:44.000513  treatment     new_page

   converted  new_page_converted  old_page_converted  intercept  ab_page
0          0                 NaN                 0.0          1        0
1          0                 0.0                 NaN          1        1
2          1                 1.0                 NaN          1        1
3          0                 NaN                 0.0          1        0
4          0                 0.0                 NaN          1        1
```

[39]:
```
display(ab_df_log_country['country'].value_counts())
```

[39]:
```
US    206364
UK     73419
CA     14695
Name: country, dtype: int64
```

[40]:
```
### Create the necessary dummy variables
ab_df_log_country[['CA', 'UK', 'US']] = pd.
 ↪get_dummies(ab_df_log_country['country'])
display(ab_df_log_country.head(5))
```

[40]:
```
   user_id country                   timestamp      group landing_page  \
0   834778      UK  2017-01-14 23:08:43.304998    control     old_page
1   928468      US  2017-01-23 14:44:16.387854  treatment     new_page
2   822059      UK  2017-01-16 14:04:14.719771  treatment     new_page
3   711597      UK  2017-01-22 03:14:24.763511    control     old_page
4   710616      UK  2017-01-16 13:14:44.000513  treatment     new_page

   converted  new_page_converted  old_page_converted  intercept  ab_page  CA  \
0          0                 NaN                 0.0          1        0   0
1          0                 0.0                 NaN          1        1   0
2          1                 1.0                 NaN          1        1   0
3          0                 NaN                 0.0          1        0   0
4          0                 0.0                 NaN          1        1   0

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

[41]:
```
ab_df_log_country['intercept'] = 1

logitmod = sm.Logit(ab_df_log_country['converted'],␣
 ↪ab_df_log_country[['intercept','ab_page', 'UK', 'US']])
results = logitmod.fit()
```

```
results.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.366238
         Iterations 6
```

[41]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              converted   No. Observations:               294478
Model:                          Logit   Df Residuals:                   294474
Method:                           MLE   Df Model:                            3
Date:                Tue, 04 May 2021   Pseudo R-squ.:               2.068e-05
Time:                        14:04:44   Log-Likelihood:            -1.0785e+05
converged:                       True   LL-Null:                   -1.0785e+05
Covariance Type:            nonrobust   LLR p-value:                    0.2158
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -2.0242      0.026    -76.696      0.000      -2.076      -1.972
ab_page       -0.0155      0.011     -1.365      0.172      -0.038       0.007
UK             0.0449      0.028      1.596      0.111      -0.010       0.100
US             0.0357      0.027      1.338      0.181      -0.017       0.088
==============================================================================
"""
```

We test for conversion of country and page above.The P-value in "US" and "UK" are 0.181 and 0.111 both are larger than 0.005,so fail to reject null hypthoese.In other word,the countries haven't effect of conversion rate.

1.h. Though we have now looked at the individual factors of country and page on conversion, we would now like to look at an interaction between page and country to see if there significant effects on conversion. Create the necessary additional columns, and fit the new model.

Provide the summary results, and our conclusions based on the results.

[42]:
```
#Create a new interaction variable between new page and country US and UK
ab_df_log_country['UK_new_page'] = ab_df_log_country['ab_page'] *␣
 ↪ab_df_log_country['UK']
ab_df_log_country['US_new_page'] = ab_df_log_country['ab_page'] *␣
 ↪ab_df_log_country['US']
```

[43]:
```
lm3 = sm.Logit(ab_df_log_country['converted'], ab_df_log_country[['intercept',␣
 ↪'ab_page', 'UK' , 'US', 'UK_new_page', 'US_new_page']])
results = lm3.fit()
results.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.366233
         Iterations 6
```

[43]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              converted   No. Observations:               294478
Model:                          Logit   Df Residuals:                   294472
Method:                           MLE   Df Model:                            5
Date:                Tue, 04 May 2021   Pseudo R-squ.:               3.438e-05
Time:                        14:04:48   Log-Likelihood:            -1.0785e+05
converged:                       True   LL-Null:                   -1.0785e+05
Covariance Type:            nonrobust   LLR p-value:                    0.1915
===============================================================================
                  coef    std err          z      P>|z|      [0.025      0.975]
-------------------------------------------------------------------------------
intercept      -1.9987      0.036    -55.323      0.000      -2.069      -1.928
ab_page        -0.0669      0.052     -1.297      0.195      -0.168       0.034
UK              0.0047      0.040      0.120      0.904      -0.073       0.082
US              0.0137      0.037      0.366      0.715      -0.060       0.087
UK_new_page     0.0809      0.056      1.436      0.151      -0.030       0.191
US_new_page     0.0444      0.053      0.832      0.405      -0.060       0.149
===============================================================================
"""
```

[44]:
```python
#exponentiated the CV to inteprete the result
np.exp(results.params)
```

[44]:
```
intercept      0.135514
ab_page        0.935326
UK             1.004759
US             1.013758
UK_new_page    1.084231
US_new_page    1.045349
dtype: float64
```

Interpretations:

1. From the above Logit Regression Results, we test for interactions of page and countries and we can see that the only intercept's p-value is less than 0.05, which is statistically significant enough for converted rate but other variables are not statistically significant.
2. The country a user lives is not statistically significant on the converted rate considering the page the user land in.
3. The user getting Converted is 1.08 times more likely to happen for UK and new page users than CA and new page users while holding all other varible constant.

4. The user getting Converted is 1.04 times more likely to happen for US and new page users than CA and new page users while holding all other varible constant.

0.4.3 Overall Conclusions and recommendation:

1. The performance of the old pages looks better as computed by different techniques.
2. So new pages couldn't bring more convesion rate and should keep the old pages.

[0]:

```

```