Analyze_ab_test_results_notebook

October 24, 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 [2]: 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 [3]: df = pd.read_csv('ab_data.csv')
        df.sample(5)
```

Out[3]:

	user_id	timestamp	group	landing_page	converted
54788	816542	2017-01-03 23:56:31.491267	control	old_page	1
17126	915998	2017-01-10 05:55:10.507561	treatment	new_page	0
213481	709142	2017-01-21 08:53:15.727334	control	old_page	0
263117	860394	2017-01-19 19:49:23.756939	treatment	new_page	0
39961	882224	2017-01-15 10:44:54.067636	treatment	new_page	0

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

```
In [4]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 294478 entries, 0 to 294477
Data columns (total 5 columns):
user_id         294478 non-null int64
timestamp       294478 non-null object
group           294478 non-null object
landing_page    294478 non-null object
converted       294478 non-null int64
dtypes: int64(2), object(3)
memory usage: 11.2+ MB
```

c. The number of unique users in the dataset.

```
In [8]: df.user_id.nunique()
```

```
Out[8]: 290584
```

d. The proportion of users converted.

```
In [17]: proportion = df.query('converted == 1')

         len(proportion)/df.shape[0]
```

```
Out[17]: 0.11965919355605512
```

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

```
In [22]: len(df.query('group == "treatment" & landing_page != "new_page"'))

         # alternatively, we can do it like this
         len(df[(df['group']=='treatment') & (df['landing_page'] != 'new_page')])
```

```
Out[22]: 1965
```

f. Do any of the rows have missing values?

```
In [23]: df.isnull().sum()

Out[23]: user_id         0
         timestamp       0
         group           0
         landing_page    0
         converted       0
         dtype: int64
```

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 [32]: # Remove the inaccurate rows, and store the result in a new dataframe df2
         df2 = df.assign(group_landing=lambda x: x.group+x.landing_page)
         df2 = df2[df2['group_landing'].isin(['controlold_page', 'treatmentnew_page'])]
         df2.drop(columns=['group_landing'], inplace=True)
         df2.sample(5)
```

```
Out[32]:        user_id                   timestamp      group landing_page  converted
         87045    779182  2017-01-17 06:50:27.662907    control     old_page          0
         172453   819017  2017-01-20 04:35:33.035952    control     old_page          0
         20815    844516  2017-01-11 05:32:27.307960  treatment     new_page          1
         29864    897165  2017-01-08 22:01:51.272491    control     old_page          0
         17040    637259  2017-01-14 01:40:49.462883  treatment     new_page          0
```

```
In [31]: # 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[31]: 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 [33]: df2.user_id.nunique()
```

```
Out[33]: 290584
```

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

```
In [39]: df2[df2.user_id.duplicated()][['user_id']]
```

```
Out[39]:       user_id
         2893   773192
```

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

```
In [40]: # confirming that the user_id is indeed duplicated
         df2.query('user_id == 773192')
```

```
Out[40]:       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 [41]: # 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.drop_duplicates(subset=['user_id'], inplace=True)
         # Check again if the row with a duplicate user_id is deleted or not
         df2[df2.user_id.duplicated()]
```

```
Out[41]: Empty DataFrame
         Columns: [user_id, timestamp, group, landing_page, converted]
         Index: []
```

```
In [42]: # seems like indeed, the duplicated row is removed and we are left with 1 unique user_i
         df2.query('user_id == 773192')
```

```
Out[42]:       user_id                   timestamp      group landing_page  converted
         1899   773192  2017-01-09 05:37:58.781806  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 [45]: # Individuals converting
         convert_individuals = len(df2.query('converted == 1'))

         # probability of individual converting
         # idea: converted + all groups (treatment + control)
         convert_individuals/df2.shape[0]
```

```
Out[45]: 0.11959708724499628
```

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

```
In [51]: # individuals in control group + converted
         converted_control = len(df2.query('converted == 1 & group == "control"'))

         # idea: how many were in control group and converted, out of the entire control group?
         control_prob = converted_control/len(df2.query('group == "control"'))
         control_prob

Out[51]: 0.1203863045004612
```

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

```
In [52]: # individuals in treatment group + converted
         converted_treatment = len(df2.query('converted == 1 & group =="treatment"'))

         # idea: how many were in treatment group and converted, out of the entire treatment gro
         treatment_prob = converted_treatment/len(df2.query('group == "treatment"'))
         treatment_prob

Out[52]: 0.11880806551510564

In [54]: # Calculate the actual difference (obs_diff) between the conversion rates for the two g
         conversation_rate_difference = abs(control_prob-treatment_prob)
         conversation_rate_difference

Out[54]: 0.0015782389853555567
```

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

```
In [56]: new_page = len(df2.query('landing_page == "new_page"'))

         # idea: new page out of all possibilities
         new_page/df2.shape[0]

Out[56]: 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.

First, we list down the probabilities of events occuring

- Probability of individual converting regardless of landing page they received: **11.96%**
- Probability of individual converting given that they are from the control group: **12.04%**
- Probability of an individual converting given that they are from the treatment group: **11.88%**
- Probability of an individual receiving the new landing page: **50%**

Based on the probabilities, control group converting is just slightly higher than treatment group, also, the probability of an individual receiving the new landing page is 50%, this means that the probability of receiving each page type is balanced. Therefore, there is insufficient evidence to conclude that the new treatment page leads to more conversions.

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.

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

H_1 : p_{old} - p_{new} < 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 [57]: # assuming that p_new and p_old are equal to the converted rate in our dataset regardle
         p_new = df2.converted.mean()
         p_new

Out[57]: 0.11959708724499628
```

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

```
In [58]: # assuming that p_new and p_old are equal to the converted rate in our dataset regardle
         p_old = df2.converted.mean()
         p_old

Out[58]: 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 [62]: n_new = df2.query('landing_page == "new_page"').count()[0]
         n_new

Out[62]: 145310
```

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

```
In [63]: n_old = df2.query('landing_page == "old_page"').count()[0]
         n_old

Out[63]: 145274
```

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

```
In [66]: # Simulate a Sample for the treatment Group
         new_page_converted = np.random.choice([0,1], n_new, p=[p_new, 1-p_new])
```

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

```
In [64]: # Simulate a Sample for the control Group
         old_page_converted = np.random.choice([0,1], n_old, p=[p_old, 1-p_old])
```

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

```
In [67]: new_mean = new_page_converted.mean()
         old_mean = old_page_converted.mean()

         diff = new_mean - old_mean
         diff
```

```
Out[67]: -0.00037642152921102401
```

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 [68]: # Sampling distribution
         p_diffs = []

         for _ in range(10000):
             new_page_converted = np.random.choice([1,0], n_new, replace=True, p=[p_new, 1-p_new
             new_mean = new_page_converted.mean()

             old_page_converted = np.random.choice([1,0], n_old, replace=True, p=[p_old, 1-p_old
             old_mean = old_page_converted.mean()

             p_diffs.append(new_mean-old_mean)
```

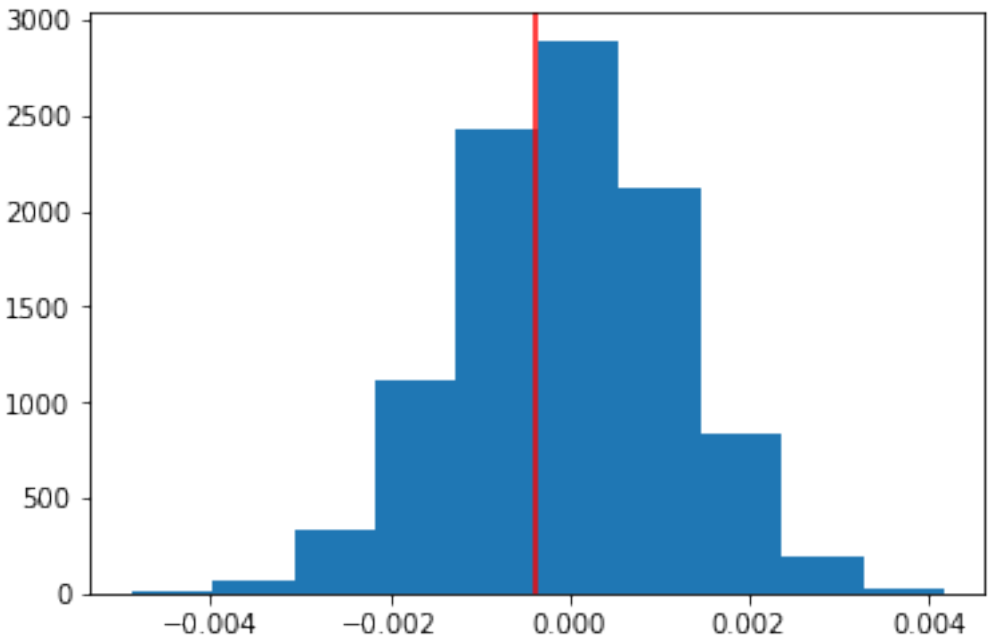
i. Histogram Plot a histogram of the **p_diffs**. Does this plot look like what you expected? Use the matching problem in the classroom to assure you fully understand what was computed here.

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

Tip: Display title, x-label, and y-label in the chart.

```
In [70]: plt.hist(p_diffs)
         plt.axvline(x=diff, color='r')
```

```
Out[70]: <matplotlib.lines.Line2D at 0x7f1efdd75f60>
```



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

```
In [71]: new_mean = df2.query('group == "treatment"')['converted'].mean()
         old_mean = df2.query('group == "control"')['converted'].mean()

         act_diff = new_mean-old_mean

         (p_diffs > act_diff).mean()
```

```
Out[71]: 0.90200000000000002
```

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 that we computed is called the p-value. If the p-value greater than the typical level of 0.05, we fail to reject the Null hypothesis. The p-value that we computed was 0.902. Therefore, we fail to reject the null hypothesis, and it seems that the old and the new pages perform almost the same.

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 [72]: import statsmodels.api as sm

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

         # number of conversions with the new_page
         convert_new = df2.query('group == "treatment"')['converted'].sum()

         # number of individuals who were shown the old_page
         n_old = df2.query('landing_page == "old_page"').count()[0]

         # number of individuals who received new_page
         n_new = df2.query('landing_page == "new_page"').count()[0]
```

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

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

The syntax is:

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

where, - `count_array` = represents the number of "converted" for each group - `nobs_array` = represents the total number of observations (rows) in each group - `alternative` = choose one of the values from `[two-sided, smaller, larger]` depending upon two-tailed, left-tailed, or right-tailed respectively. >**Hint**: It's a two-tailed if you defined H_1 as $(p_{new} = p_{old})$. It's a left-tailed if you defined H_1 as $(p_{new} < p_{old})$. It's a right-tailed if you defined H_1 as $(p_{new} > p_{old})$.

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

```
In [73]: import statsmodels.api as sm
         # ToDo: Complete the sm.stats.proportions_ztest() method arguments
         z_score, p_value = sm.stats.proportions_ztest([convert_old, convert_new], [n_old, n_new
         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.**?

```
In [76]: from scipy.stats import norm

         # assuming 95% confidence interval
         norm.ppf(1-0.05)
```

```
Out[76]: 1.6448536269514722
```

> The z_score we computed is 1.3109 which is less than 1.64485, and the p-value is different than the findings in parts j. and k. Therefore, we fail to reject the null hypothesis, and our conclusion is the same as in part k.

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 are looking for a binary outcome of conversion or no conversion.

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 [77]: # Adding an intercept column
         df2['intercept'] = 1

         # Creating a dummy variable column
         df2['ab_page'] = pd.get_dummies(df2['group'])['treatment']

         # To check
         df2.head()
```

Out[77]:

	user_id	timestamp	group	landing_page	converted	intercept	ab_page
0	851104	2017-01-21 22:11:48.556739	control	old_page	0	1	0
1	804228	2017-01-12 08:01:45.159739	control	old_page	0	1	0
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0	1	1
3	853541	2017-01-08 18:28:03.143765	treatment	new_page	0	1	1
4	864975	2017-01-21 01:52:26.210827	control	old_page	1	1	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 [78]: # Instantiating regression model
         logit_model = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])

         # Fitting the model
         results = logit_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 [79]: from scipy import stats
         stats.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df)

         results.summary()
```

```
Out[79]: <class 'statsmodels.iolib.summary.Summary'>
         """
                                   Logit Regression Results
         ==============================================================================
```

```
Dep. Variable:              converted   No. Observations:              290584
Model:                         Logit   Df Residuals:                  290582
Method:                          MLE   Df Model:                           1
Date:               Mon, 24 Oct 2022   Pseudo R-squ.:               8.077e-06
Time:                       08:54:36   Log-Likelihood:            -1.0639e+05
converged:                      True   LL-Null:                   -1.0639e+05
                                       LLR p-value:                    0.1899
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -1.9888      0.008   -246.669      0.000      -2.005      -1.973
ab_page       -0.0150      0.011     -1.311      0.190      -0.037       0.007
==============================================================================
"""
```

e. What is the p-value associated with **ab_page**? Why does it differ from the value you found in **Part II**?

The p-value associated with ab_page is 0.190, it differs from the p-value that we derived in Part II because they have differing explanatory variables.

The null and alternative hypothesis for our regression model are:

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

H_1 : p_{old} - p_{new} != 0

The null and alternative hypotheses in Part II was a one-sided test, where the null and alternative hypotheses for the regression model, is two-sided test.

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?

It is a good idea to consider other variables to add to the regression model as it may contribute to the significance of the results and lead to more accurate decisions. However, even with additional factors, we still can not consider all influencing factors. So, it may lead to correlated errors and multicollinearity.

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

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

2. Does it appear that country had an impact on conversion? To answer this question, consider the three unique values, `['UK', 'US', 'CA']`, in the `country` column. Create dummy variables for these country columns. >**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 [80]: # Read the countries.csv
         countries = pd.read_csv('countries.csv')
         countries.sample(5)
```

Out[80]:
```
        user_id country
221785   939457      UK
216593   769737      UK
81585    913831      US
139442   687557      US
287515   874894      UK
```

In [82]:
```
# Join with the df2 dataframe
df3 = df2.merge(countries, on='user_id', how='inner')
df3.sample(5)
```

Out[82]:
```
        user_id                   timestamp      group landing_page  \
76145    672468  2017-01-24 13:24:42.574067    control     old_page
8012     719131  2017-01-10 16:25:00.743527  treatment     new_page
253164   840821  2017-01-13 07:22:11.161009    control     old_page
144811   886400  2017-01-09 03:10:53.081231  treatment     new_page
137993   637183  2017-01-11 18:42:10.353909    control     old_page

        converted  intercept  ab_page country
76145           0          1        0      US
8012            0          1        1      UK
253164          0          1        0      US
144811          0          1        1      UK
137993          0          1        0      UK
```

In [84]:
```
df3.country.unique()
```

Out[84]:
```
array(['US', 'CA', 'UK'], dtype=object)
```

In [85]:
```
# Create the necessary dummy variables
df3[['US', 'CA', 'UK']] = pd.get_dummies(df3['country'])
df3.sample(5)
```

Out[85]:
```
        user_id                   timestamp      group landing_page  \
132827   941912  2017-01-10 09:02:53.829681    control     old_page
258231   739514  2017-01-07 20:13:42.993698  treatment     new_page
153841   769364  2017-01-19 08:43:35.355530    control     old_page
226683   859783  2017-01-07 15:57:48.273194  treatment     new_page
272630   693603  2017-01-14 20:21:51.659866    control     old_page

        converted  intercept  ab_page country  US  CA  UK
132827          0          1        0      US   0   0   1
258231          0          1        1      US   0   0   1
153841          0          1        0      UK   0   1   0
226683          0          1        1      UK   0   1   0
272630          1          1        0      US   0   0   1
```

In [86]:
```
# Fit your model, and summarize the results
df3['intercept'] = 1
```

```
logit2 = sm.Logit(df3['converted'], df3[['intercept', 'UK', 'US']])
res2 = logit2.fit()
res2.summary()
```

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

Out[86]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              converted   No. Observations:               290584
Model:                          Logit   Df Residuals:                   290581
Method:                           MLE   Df Model:                            2
Date:                Mon, 24 Oct 2022   Pseudo R-squ.:               1.521e-05
Time:                        09:10:02   Log-Likelihood:            -1.0639e+05
converged:                       True   LL-Null:                   -1.0639e+05
                                        LLR p-value:                    0.1984
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -1.9868      0.011   -174.174      0.000      -2.009      -1.964
UK            -0.0099      0.013     -0.746      0.456      -0.036       0.016
US            -0.0507      0.028     -1.786      0.074      -0.106       0.005
==============================================================================
"""
```

In [87]:
```
np.exp(res2.params)
```

Out[87]:
```
intercept    0.137132
UK           0.990133
US           0.950546
dtype: float64
```

In [88]:
```
1/np.exp(res2.params)
```

Out[88]:
```
intercept    7.292253
UK           1.009966
US           1.052027
dtype: float64
```

Based on the statistical output:

- If an individual lives in US, they are 0.99 times more likely to be converted while holding all other variables constant.
- If an individual lives in UK, they are 0.95 times more likely to be converted while holding all other variables constant.

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 [90]: df3['UK_page'] = df3['ab_page']*df3['UK']
         df3['US_page'] = df3['ab_page']*df3['US']

         df3.head()
```

Out[90]:

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

	intercept	ab_page	country	US	CA	UK	UK_page	US_page
0	1	0	US	0	0	1	0	0
1	1	0	US	0	0	1	0	0
2	1	1	US	0	0	1	1	0
3	1	1	US	0	0	1	1	0
4	1	0	US	0	0	1	0	0

```
In [91]: logit3 = sm.Logit(df3['converted'], df3[['intercept', 'UK', 'US', 'ab_page', 'UK_page',
         res3 = logit3.fit()
         res3.summary()

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

Out[91]:

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              converted   No. Observations:               290584
Model:                          Logit   Df Residuals:                   290578
Method:                           MLE   Df Model:                            5
Date:                Mon, 24 Oct 2022   Pseudo R-squ.:               3.482e-05
Time:                        09:15:05   Log-Likelihood:            -1.0639e+05
converged:                       True   LL-Null:                   -1.0639e+05
                                        LLR p-value:                    0.1920
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
```

```
intercept     -1.9922      0.016   -123.457      0.000      -2.024      -1.961
UK             0.0057      0.019      0.306      0.760      -0.031       0.043
US            -0.0118      0.040     -0.296      0.767      -0.090       0.066
ab_page        0.0108      0.023      0.475      0.635      -0.034       0.056
UK_page       -0.0314      0.027     -1.181      0.238      -0.084       0.021
US_page       -0.0783      0.057     -1.378      0.168      -0.190       0.033
==============================================================================
"""
```

```
In [92]: np.exp(res3.params)
```

```
Out[92]: intercept    0.136392
         UK           1.005761
         US           0.988285
         ab_page      1.010893
         UK_page      0.969090
         US_page      0.924703
         dtype: float64
```

```
In [93]: 1/np.exp(res3.params)
```

```
Out[93]: intercept    7.331806
         UK           0.994272
         US           1.011854
         ab_page      0.989224
         UK_page      1.031896
         US_page      1.081428
         dtype: float64
```

Results: Since the p-values exceed 0.05, None of them are significant. Therefore, we fail to reject the null hypothesis, And we conclude that there is not sufficient evidence that there is an interaction between the country and the page received that predicts whether or not the user is converting.

1.1 Summary

The old and new pages have similar performance based on our statistical tests conducted. Therefore, the new page should not be implemented as there is insufficient evidence to demonstrate that the new page will outperform the old page.

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