Analyze_ab_test_results_notebook

September 24, 2023

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.
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') #to read in our data file
        df.head() #to make sure our code worked and see the first few rows
```

```
Out[3]:    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 [4]: df.count() #gives us the count of the rows
        #there are 294478 rows in the dataset
```

```
Out[4]: user_id         294478
        timestamp       294478
        group           294478
        landing_page    294478
        converted       294478
        dtype: int64
```

c. The number of unique users in the dataset.

```
In [5]: df.nunique() #gives us the amount of unique values in every column
        #there are 290584 unique users in the dataset
```

```
Out[5]: user_id         290584
        timestamp       294478
        group                2
        landing_page         2
        converted            2
        dtype: int64
```

d. The proportion of users converted.

```
In [6]: mean1 = df.converted.mean()#gives the mean of users that converted
        '{:.3%}'.format(mean1) #outputs the mean in percentage form
```

```
Out[6]: '11.966%'
```

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

```
In [7]: df.query("group == 'treatment' and landing_page != 'new_page'").count()
        #counts the number of rows where these two stipulations are present
        #it occurs 1965 times
```

```
Out[7]: user_id         1965
        timestamp       1965
        group           1965
        landing_page    1965
        converted       1965
        dtype: int64
```

f. Do any of the rows have missing values?

```
In [8]: df.info() #this displays the number of non-null values, where null would mean missing
        #we can see that every column has the full amount of entries so there are no missing val
```

```
<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
```

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 [9]: # Remove the inaccurate rows, and store the result in a new dataframe df2
        #code below lays out all the conditions that are found in inaccurate rows and gives us t
        delete = df.query('(group == "treatment" and landing_page != "new_page") or (group != "t
        df2 = df.drop(delete) #we use those index numbers to tell the program to drop those inac
```

```
In [10]: # 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[10]: 0
```

1.0.3 ToDo 1.3

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

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

```
In [11]: df2.nunique() #gives us number of unique values in each column
         #there are 290584 unique values
```

```
Out[11]: user_id         290584
         timestamp       290585
         group                2
         landing_page         2
         converted            2
         dtype: int64
```

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

```
In [12]: df2[df2.duplicated('user_id', keep=False)] #shows the rows where the same user id appea
         #the user_id repeated is 773192
```

```
Out[12]:       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
```

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

```
In [13]: df2[df2.duplicated('user_id', keep=False)] #shows the rows where the same user id appea
```

```
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]: # Remove one of the rows with a duplicate user_id..
         df2 = df2.drop(1899) #drops one of the duplicates based off it's index #
         # Check again if the row with a duplicate user_id is deleted or not
         df2.query('user_id == 773192') #we can see only one row with the user id now so it work
```

```
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() #gives us the mean of the converted column
```

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

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

```
In [16]: df2[df2['group'] == 'control'].converted.mean() #chooses the rows with the control grou
```

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

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

```
In [17]: df2[df2['group'] == 'treatment'].converted.mean() #chooses the rows with the treatment
```

```
Out[17]: 0.11880806551510564
```

```
In [18]: # Calculate the actual difference (obs_diff) between the conversion rates for the two g
         b = 0.1204 #rate in part b
         c = 0.1188 #rate in part c
         obs_diff = c - b #the difference
```

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

```
In [19]: users = 290584 #number of entries or unique users in the data file
         m = df2.query('landing_page == "new_page"')['user_id'].nunique() #number of rows where
         m / users #division to give us the mean
```

```
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.

> Considering that the probability of an individual converting regardless of the page they receive, ~0.12, is similar to both the probabilities of conversion to those in the control and treatment groups, there is no statistical significance to find that the new treatment group led to more conversions. This is further supported by a difference of 0 when subtracting their respective probabilities and an overall 50% chance of an individual receiving the new page regardless.

## 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.
$H_0 = p_{old} \geq p_{new}$
$H_1 = p_{old} < p_{new}$

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_new = df2.converted.mean() #calculates the overall conversion rate

         p_new
```

```
Out[20]: 0.11959708724499628
```

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

```
In [21]: p_old = p_new #since under the null we assume these values are the same
         p_old
```

```
Out[21]: 0.11959708724499628
```

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

```
In [22]: n_new = df2[df2['group'] == 'treatment']['user_id'].count() #counts the number of users
         n_new
```

```
Out[22]: 145310
```

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

```
In [23]: n_old = df2[df2['group'] == 'control']['user_id'].count()  #counts the number of users
         n_old
```

```
Out[23]: 145274
```

e. Simulate Sample for the `treatment` Group Simulate n_{new} transactions with a conversion rate of p_{new} under the null hypothesis. Store these n_{new} 1's and 0's in the `new_page_converted` numpy array.

```
In [24]: new_page_converted = np.random.choice([0, 1], size=n_new, p=[(1 - p_new), 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 [25]: old_page_converted = np.random.choice([0, 1], size=n_old, p=[(1 - p_old), 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 [26]: con_diff = new_page_converted.mean() - old_page_converted.mean()
         con_diff

Out[26]: 0.00052107827970916676
```

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.

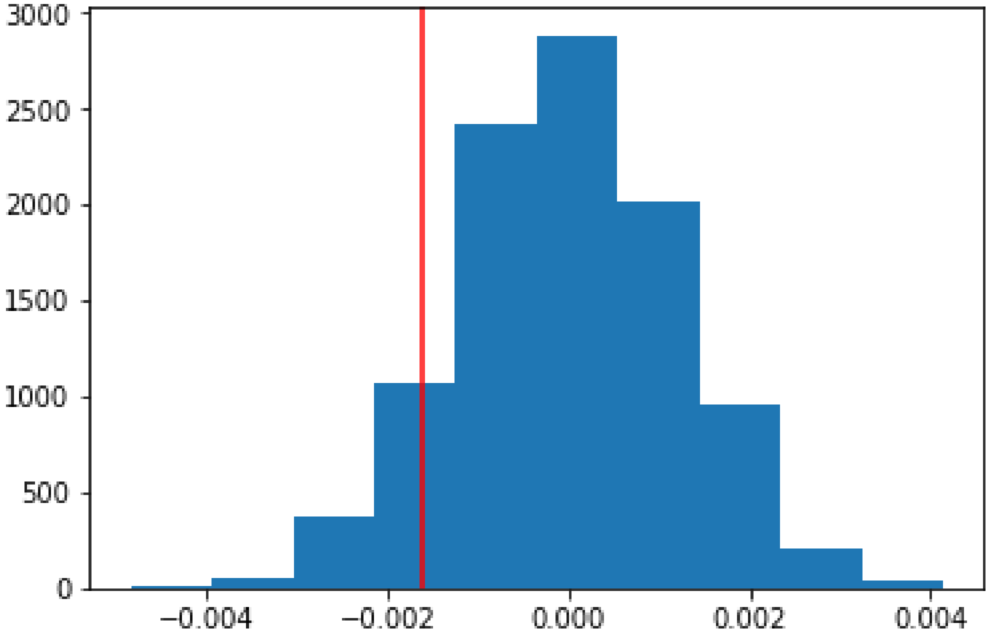
Store all $(p'_{new}$ - $p'_{old})$ values in a NumPy array called `p_diffs`.

```
In [27]: # Sampling distribution
         p_diffs = []
         size = df2.shape[0]
         for _ in range(10000):
             new_page_converted = np.random.choice([0, 1], size=n_new, p=[(1 - p_new), p_new])
             old_page_converted = np.random.choice([0, 1], size=n_new, p=[(1 - p_old), p_old])
             con_diff = new_page_converted.mean() - old_page_converted.mean()
             p_diffs.append(con_diff)
```

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 [28]: plt.hist(p_diffs);
         plt.axvline(obs_diff, c='red');
```



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

```
In [29]: (np.array(p_diffs) > obs_diff).mean()
```

```
Out[29]: 0.9076999999999995
```

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 p-value was computed, which is the probability of getting our statistic or a more extreme value if the null is true. Essentially we computed the probability that the difference in our conversion rates from the sample data is greater than the actual difference in our conversion rates. Our value is around ~0.9 which is higher than our Type I error rate of 0.05. This means we fail to reject the null or put more simply our calculations find the null hypothesis to be true.

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

```
        # number of conversions with the old_page
        convert_old = df2.query('converted == 1 and landing_page == "old_page"')['user_id'].cou

        # number of conversions with the new_page
        convert_new = df2.query('converted == 1 and landing_page == "new_page"')['user_id'].cou

        # number of individuals who were shown the old_page
        n_old = df2[df2['group'] == 'control']['user_id'].count()

        # number of individuals who received new_page
        n_new = df2[df2['group'] == 'treatment']['user_id'].count()
```

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

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

The syntax is:

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

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

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

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

```
In [32]: count = np.array([convert_old, convert_new])
         nobs = np.array([n_old, n_new])
         sm.stats.proportions_ztest(count, nobs, alternative='smaller')
         #returns z-score and p-value
```

```
Out[32]: (1.3109241984234394, 0.90505831275902449)
```

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.**?

> The z-score represents the distance between the two "converted" success rates in terms of the standard error. The distance would need to be larger than ~1.64 for our statistics to be significant enough to reject to the null, or accept that our findings show that the new page converts users better than the old page. Our z-score is less than the value needed, alongside the similar p-value from our previous computions in part j that had us fail to rejecct the null. So accepting the null is the safest option.

### 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

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 [33]: df2['intercept'] = 1
         df2['ab_page'] = pd.get_dummies(df2['group'])['treatment']
         df2.head()
```

```
Out[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

            intercept  ab_page
         0          1        0
         1          1        0
         2          1        1
         3          1        1
         4          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 [34]: import statsmodels.api as sm
         logit_mod = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])
```

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

```
In [35]: results = logit_mod.fit()
         results.summary2()
```

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

```
Out[35]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                  Results: Logit
         ==================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2023-08-07 01:09 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**?

Hints: - What are the null and alternative hypotheses associated with your regression model, and how do they compare to the null and alternative hypotheses in **Part II**? - You may comment on if these hypothesis (Part II vs. Part III) are one-sided or two-sided. - You may also compare the current p-value with the Type I error rate (0.05).

> The p-value for ab_page is 0.1899 compared to the 0.9 value we found in Part II. It is important to note here that Part II was used to find the p-value under the null hypothesis, meaning we are assuming that the new page does not statistically influence the conversion rate and calculating how likely that is. We found no statistical evidence to reject our theory, the null hypothesis(the conversion rate was the same if not better than the old page). Whereas for Part III we found our p-value without assuming our null to be true. We let the data speak for itself instead of using a sample. Another detail to consider is that the part II tests were one sided, meaning we are interpreting the results from a specific direction (alternative is more than the null rate). The part III tests were two sided, meaning the direction was not specified (the alternative is different from the null). We still found that our value is larger than the type I error rate which means 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?

> Adding other factors helps to make our statistics more encompassing, accurate, and realistic. By considering additional factors we can use our statistics to make more educated decisions. Still, the drawback is identifying which factors are worth considering since too many can make our data confusing to interpret and potentially lessen the

accuracy. We could find that some factors are more related to each other than they are the results they are meant to influence

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 [36]: # Read the countries.csv
         countries = pd.read_csv('countries.csv') #to read in our data file
         countries.head() #to make sure our code worked and see the first few rows
```

```
Out[36]:    user_id country
         0   834778      UK
         1   928468      US
         2   822059      UK
         3   711597      UK
         4   710616      UK
```

```
In [37]: # Join with the df2 dataframe
         df_merged = df2.set_index('user_id').join(countries.set_index('user_id'))
         df_merged.head()
```

```
Out[37]:                           timestamp      group landing_page  converted  \
         user_id
         851104   2017-01-21 22:11:48.556739    control     old_page          0
         804228   2017-01-12 08:01:45.159739    control     old_page          0
         661590   2017-01-11 16:55:06.154213  treatment     new_page          0
         853541   2017-01-08 18:28:03.143765  treatment     new_page          0
         864975   2017-01-21 01:52:26.210827    control     old_page          1

                  intercept  ab_page country
         user_id
         851104           1        0      US
         804228           1        0      US
         661590           1        1      US
         853541           1        1      US
         864975           1        0      US
```

```
In [38]: df_merged.country.value_counts() #so I can see the different types of values in this co
```

```
Out[38]: US    203619
         UK     72466
         CA     14499
         Name: country, dtype: int64
```

```
In [39]: df_merged[['CA', 'UK', 'US']] = pd.get_dummies(df_merged['country'])
         df_merged.head()
```

```
Out[39]:                           timestamp      group landing_page  converted  \
         user_id
         851104   2017-01-21 22:11:48.556739    control     old_page          0
         804228   2017-01-12 08:01:45.159739    control     old_page          0
         661590   2017-01-11 16:55:06.154213  treatment     new_page          0
         853541   2017-01-08 18:28:03.143765  treatment     new_page          0
         864975   2017-01-21 01:52:26.210827    control     old_page          1

                  intercept  ab_page country  CA  UK  US
         user_id
         851104           1        0      US   0   0   1
         804228           1        0      US   0   0   1
         661590           1        1      US   0   0   1
         853541           1        1      US   0   0   1
         864975           1        0      US   0   0   1
```

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.

> **Hints**: - Look at all of p-values in the summary, and compare against the Type I error rate (0.05). - Can you reject/fail to reject the null hypotheses (regression model)? - Comment on the effect of page and country to predict the conversion.

```
In [46]: df_merged['UK_ab'] = df_merged['UK']*df_merged['ab_page']
         df_merged['CA_ab'] = df_merged['CA']*df_merged['ab_page']
```

```
In [48]: # Fit your model, and summarize the results
         df_merged['intercept'] = 1
         logit_mod = sm.Logit(df_merged['converted'], df_merged[['intercept', 'ab_page', 'UK', '
         results = logit_mod.fit()
         results.summary2()
```

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

```
Out[48]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                  Results: Logit
         ==================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2023-08-07 01:41 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           -0.0057    0.0188    -0.3057  0.7598  -0.0426   0.0311
         CA           -0.0175    0.0377    -0.4652  0.6418  -0.0914   0.0563
         UK_ab         0.0314    0.0266     1.1807  0.2377  -0.0207   0.0835
         CA_ab        -0.0469    0.0538    -0.8718  0.3833  -0.1523   0.0585
         ==================================================================

         """
```

```
In [49]: np.exp(results.params)
```

```
Out[49]: intercept    0.137178
         ab_page      0.979646
         UK           0.994272
         CA           0.982625
         UK_ab        1.031896
         CA_ab        0.954198
         dtype: float64
```

```
In [50]: 1/np.exp(results.params)
```

```
Out[50]: intercept    7.289813
         ab_page      1.020776
         UK           1.005761
         CA           1.017682
         UK_ab        0.969090
         CA_ab        1.048001
         dtype: float64
```

Looking at our p-values we can see that each are more than the type I error rate, meaning we cannot reject the null hypotheses based solely on these values. Our data shows that those in the US are 0.9% more likely to convert than in the UK and 0.9% more likely to convert than in Canada. With our combined columns of country and page, we see fairly similar ~1% values meaning there is a very small effect on conversion

> when taking country into consideration. I would not consider this data to be statiscally significant. If we look at this practically it still seems that we have no statistical basis on which to reject the null.

## 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!

> **Tip**: Once you are satisfied with your work here, check over your notebook to make sure that it satisfies all the specifications mentioned in the rubric. You should also probably remove all of the "Hints" and "Tips" like this one so that the presentation is as polished as possible.

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

Out[51]: 0

In [ ]:
```