Analyze_ab_test_results_notebook

October 1, 2021

1 Analyze A/B Test Results

This project will assure you have mastered the subjects covered in the statistics lessons. We have organized the current notebook into the following sections:

- Section **??**
- Section **??**
- Section **??**
- Section **??**
- Section **??**
- Section **??**

Specific programming tasks are marked with a **ToDo** tag.

Introduction

A/B tests are very commonly performed by data analysts and data scientists. For this project, you will be working to understand the results of an A/B test run by an e-commerce website. Your goal is to work through this notebook to help the company understand if they should: - Implement the new webpage, - Keep the old webpage, or - Perhaps run the experiment longer to make their decision.

Each **ToDo** task below has an associated quiz present in the classroom. Though the classroom quizzes are **not necessary** to complete the project, they help ensure you are on the right track as you work through the project, and you can feel more confident in your final submission meeting the rubric specification.

Part I - Probability

To get started, let's import our libraries.

```
In [1]: import 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 [2]: df = pd.read_csv('ab_data.csv')
        df.head()
```

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

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

```
In [3]: df.shape
```

```
Out[3]: (294478, 5)
```

c. The number of unique users in the dataset.

```
In [4]: df['user_id'].nunique()
```

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

d. The proportion of users converted.

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

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

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

```
In [6]: df.query('group == "treatment"')['landing_page'].value_counts()
```

```
Out[6]: new_page    145311
        old_page      1965
        Name: landing_page, dtype: int64
```

```
In [7]: df.query('group == "control"')['landing_page'].value_counts()
```

```
Out[7]: old_page    145274
        new_page      1928
        Name: landing_page, dtype: int64
```

f. Do any of the rows have missing values?

```
In [8]: df.isna().sum()
```

```
Out[8]: user_id         0
        timestamp       0
        group           0
        landing_page    0
        converted       0
        dtype: int64
```

1.0.2 ToDo 1.2

In a particular row, the **group** and **landing_page** columns should have either of the following acceptable values:

user_id	timestamp	group	landing_page	converted
XXXX	XXXX	`control`	`old_page`	X
XXXX	XXXX	`treatment`	`new_page`	X

It means, the `control` group users should match with `old_page`; and `treatment` group users should matched with the `new_page`.

However, for the rows where `treatment` does not match with `new_page` or `control` does not match with `old_page`, we cannot be sure if such rows truly received the new or old wepage.

Use **Quiz 2** in the classroom to figure out how should we handle the rows where the group and landing_page columns don't match?

a. Now use the answer to the quiz to create a new dataset that meets the specifications from the quiz. Store your new dataframe in **df2**.

```
In [9]: # Remove the inaccurate rows, and store the result in a new dataframe df2
        df2 = df[((df['group'] == 'treatment')& (df['landing_page'] == 'new_page'))]
        df2 = df2.append(df[((df['group'] == 'control')& (df['landing_page'] == 'old_page'))],ig
        df2.shape

Out[9]: (290585, 5)
```

```
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.user_id.nunique()

Out[11]: 290584
```

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

```
In [12]: df2[df2.duplicated(subset='user_id',keep=False)]

Out[12]:       user_id                   timestamp      group landing_page  converted
         938    773192  2017-01-09 05:37:58.781806  treatment     new_page          0
         1404   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(subset='user_id',keep=False)]

Out[13]:       user_id                   timestamp      group landing_page  converted
         938    773192  2017-01-09 05:37:58.781806  treatment     new_page          0
         1404   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..
         # Hint: The dataframe.drop_duplicates() may not work in this case because the rows with
         df2.drop_duplicates(subset=['user_id'],inplace=True)
         df2[df2.user_id == 773192]
         # Check again if the row with a duplicate user_id is deleted or not
```

```
Out[14]:     user_id                   timestamp      group landing_page  converted
         938  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 [15]: df2.converted.mean()
```

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

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

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

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

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

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

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

```
In [18]: # Calculate the actual difference (obs_diff) between the conversion rates for the two g
         obs_diff = df2.query('group == "treatment"')['converted'].mean() - df2.query('group ==
         obs_diff
```

```
Out[18]: -0.0015782389853555567
```

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

```
In [19]: df2.landing_page.value_counts(normalize=True)
```

```
Out[19]: new_page    0.500062
         old_page    0.499938
         Name: landing_page, dtype: float64
```

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

In this moment it's looks like new_page has lower conversion rate then the old_page.

Part II - A/B Test

Since a timestamp is associated with each event, you could run a hypothesis test continuously as long as you observe the events.

However, then the hard questions would be: - Do you stop as soon as one page is considered significantly better than another or does it need to happen consistently for a certain amount of time?
- How long do you run to render a decision that neither page is better than another?

These questions are the difficult parts associated with A/B tests in general.

1.0.5 ToDo 2.1

For now, consider you need to make the decision just based on all the data provided.

If you want to assume that the old page is better unless the new page proves to be definitely better at a Type I error rate of 5%, what should be your null and alternative hypotheses (H_0 and H_1)?

You can state your hypothesis in terms of words or in terms of p_{old} and p_{new}, which are the "converted" probability (or rate) for the old and new pages respectively.

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

$$H_1 : p_{old} - p_{new} < 0$$

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 [20]: df2['converted'].mean()
```

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

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

```
In [21]: df2['converted'].mean()
```

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

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

```
In [22]: df2.query('group == "treatment"').shape[0]
```

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

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

```
In [23]: df2.query('group == "control"').shape[0]
```

```
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. *Hint*: Use `numpy.random.choice()` method to randomly generate n_{new} number of values. Store these n_{new} 1's and 0's in the `new_page_converted` numpy array.

```
In [24]: # Simulate a Sample for the treatment Group
         new_page_converted =np.random.choice(df2.query('group == "treatment"')['converted'],df2
```

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

```
In [26]: # Simulate a Sample for the control Group
         old_page_converted =np.random.choice(df2.query('group == "control"')['converted'],df2.q
```

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

```
In [27]: new_page_converted.mean() - old_page_converted.mean()
```

```
Out[27]: 7.3468842708951376e-05
```

h. Sampling distribution Re-create `new_page_converted` and `old_page_converted` and find the $(p'_{new} - p'_{old})$ value 10,000 times using the same simulation process you used in parts (a) through (g) above.

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

```
In [28]: # Sampling distribution
         p_diffs, new_page_conv_rate, old_page_conv_rate = [], [],[]
         for i in range(10000):
             new_page_converted =np.random.choice(df2.query('group == "treatment"')['converted']
             old_page_converted =np.random.choice(df2.query('group == "control"')['converted'],d
```

```
        p_diffs.append(new_page_converted.mean() - old_page_converted.mean())
        new_page_conv_rate.append(new_page_converted.mean())
        old_page_conv_rate.append(old_page_converted.mean())
    p_diffs = np.array(p_diffs)
    new_page_conv_rate = np.array(new_page_conv_rate)
    old_page_conv_rate = np.array(old_page_conv_rate)

    p_diffs.mean(), old_page_conv_rate.mean(), new_page_conv_rate.mean()
```

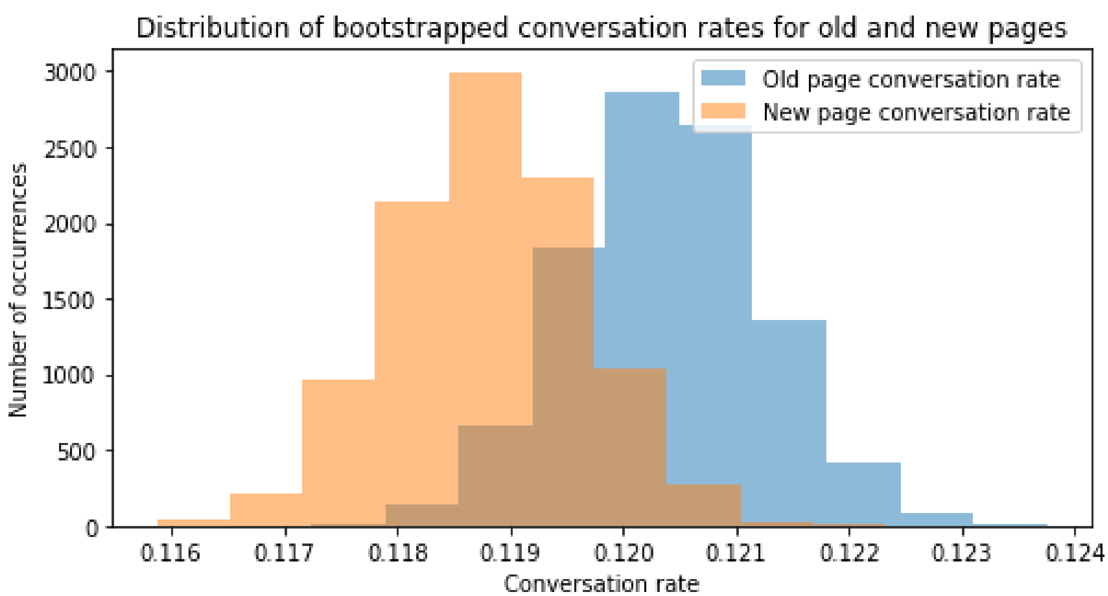
Out[28]: (-0.0015792703782423071, 0.12038551839971365, 0.11880624802147133)

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.

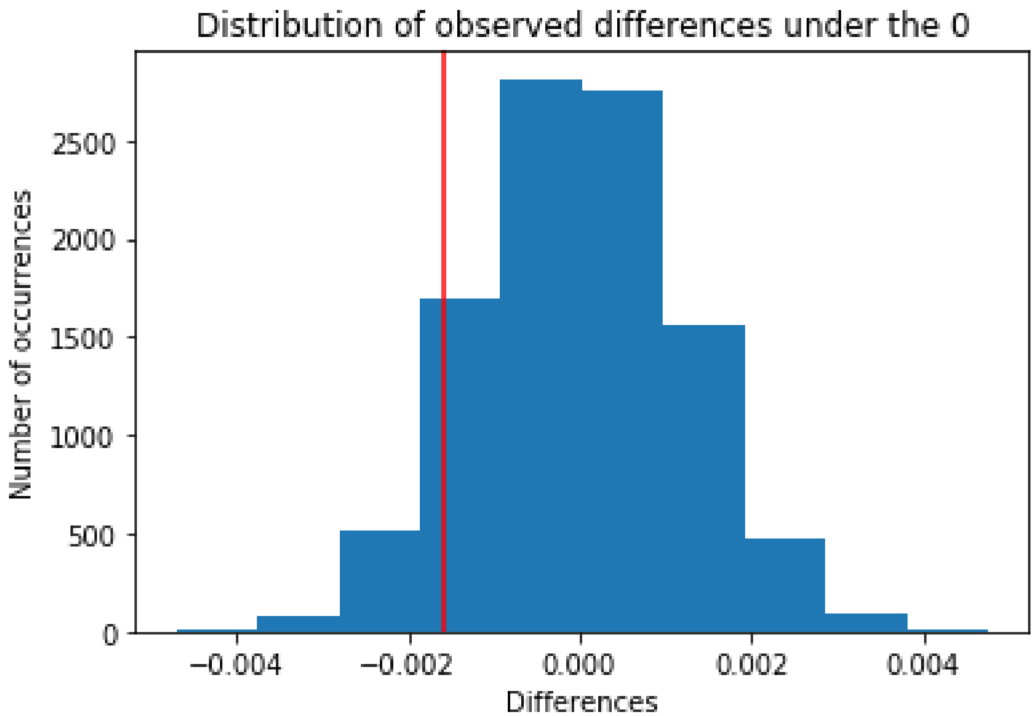
Tip: Display title, x-label, and y-label in the chart.

In [30]:
```
plt.figure(figsize=(8,4));
plt.hist(old_page_conv_rate, alpha = 0.5, label='Old page conversation rate');
plt.hist(new_page_conv_rate, alpha = 0.5, label='New page conversation rate');
plt.legend()
plt.title('Distribution of bootstrapped conversation rates for old and new pages')
plt.xlabel('Conversation rate');
plt.ylabel('Number of occurrences');
```



In [32]:
```
null_vals = np.random.normal(0,p_diffs.std(),p_diffs.size)
```

```
In [33]: plt.hist(null_vals);
         plt.axvline(obs_diff,color = 'r');
         plt.title('Distribution of observed differences under the 0');
         plt.xlabel('Differences');
         plt.ylabel('Number of occurrences');
```



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

```
In [40]: (null_vals > obs_diff).mean()
```

```
Out[40]: 0.9029000000000004
```

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)".

Above, we found the probability of a type 1 error (false positive) in H_1. This means that hypothesis # 1 does not work in about 90.14% of cases.

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 [45]: 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('group == "control"')['converted'].count()

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

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

The syntax is:

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

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

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

```
In [46]: 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],a
         print(z_score, p_value)

1.31092419842 0.905058312759
```

n. What do the z-score and p-value you computed in the previous question mean for the conversion rates of the old and new pages? Do they agree with the findings in parts **j.** and **k.**?

Z-score is a numerical measurement that describes a value's relationship to the mean of a group of values.

While we defining the hypothesis we assume that H_0 hypothesis is always true, so in the testing above with p-value over 0.9 we can easily reject H_1.

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 [47]: df2['intercept'] = 1
         df2 = df2.join(pd.get_dummies(df2['group']))
         df2.drop(columns='control',inplace=True)
         df2.rename(columns={'treatment':'ab_page'},inplace=True)
         df2.head()
```

```
Out[47]:    user_id                   timestamp      group landing_page  converted  \
         0   661590  2017-01-11 16:55:06.154213  treatment     new_page          0
         1   853541  2017-01-08 18:28:03.143765  treatment     new_page          0
         2   679687  2017-01-19 03:26:46.940749  treatment     new_page          1
         3   817355  2017-01-04 17:58:08.979471  treatment     new_page          1
         4   839785  2017-01-15 18:11:06.610965  treatment     new_page          1

            intercept  ab_page
         0          1        1
         1          1        1
         2          1        1
         3          1        1
         4          1        1
```

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 [48]: log_mod = sm.Logit(df2.converted,df2[['intercept','ab_page']])
         results = log_mod.fit()
         print(results.summary2())
```

```
Optimization terminated successfully.
         Current function value: 0.366118
         Iterations 6
                          Results: Logit
=================================================================
Model:              Logit            No. Iterations:   6.0000
Dependent Variable: converted        Pseudo R-squared: 0.000
Date:               2021-10-01 19:15 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
=================================================================
```

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

```
In [49]: 1/np.exp(-0.0150)
```

```
Out[49]: 1.0151130646157189
```

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

H_0 - the probability that users will be converted when we show them an old page a the same that if we show them a new page.

H_1 - the probability that users will be converted when we show them a new page a higher o lower, then if we show them an old page.

P-value associated with ab_page: 0.1899. It differs from previous parts in case that in logistic regression we have done the two-tailed test.

Coefficient explain to us that if we show to user a new page, the chances that he will become converted lowering by 1.015 times.

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?

We could try to add timestamps in our analysis. So I divide the experiment into 2 parts, the first part is coded by 0, and the second part is coded by 1.

```
In [50]: df2.timestamp.min(), df2.timestamp.max()
```

```
Out[50]: ('2017-01-02 13:42:05.378582', '2017-01-24 13:41:54.460509')
```

```
In [51]: df2['half_exp'] = df2['timestamp'].apply(lambda x: 1 if x >= '2017-01-13' else 0)
         log_mod = sm.Logit(df2.converted,df2[['intercept','ab_page','half_exp']])
         results = log_mod.fit()
         print(results.summary2())
```

```
Optimization terminated successfully.
         Current function value: 0.366117
         Iterations 6
                          Results: Logit
=================================================================
Model:              Logit            No. Iterations:   6.0000
Dependent Variable: converted        Pseudo R-squared: 0.000
Date:               2021-10-01 19:15 AIC:              212781.2513
No. Observations:   290584           BIC:              212812.9903
Df Model:           2                Log-Likelihood:   -1.0639e+05
Df Residuals:       290581           LL-Null:          -1.0639e+05
Converged:          1.0000           Scale:            1.0000
```

```
-----------------------------------------------------------------
               Coef.   Std.Err.      z      P>|z|    [0.025   0.975]
-----------------------------------------------------------------
intercept    -1.9951    0.0101  -197.9198  0.0000  -2.0149  -1.9753
ab_page      -0.0150    0.0114    -1.3085  0.1907  -0.0374   0.0074
half_exp      0.0120    0.0115     1.0482  0.2946  -0.0104   0.0344
=================================================================
```

If we looking at logistic regression results I think we should try to continue the experiment to collect more data or add some more variables that don't exist in our current data.

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 [52]:
```
# Read the countries.csv
countries_df = pd.read_csv('countries.csv')
```

In [53]:
```
# Join with the df2 dataframe
df_new = countries_df.set_index('user_id').join(df2.set_index('user_id'), how='inner')
```

In [54]:
```
# Create the necessary dummy variables
df_new = df_new.join(pd.get_dummies(df_new['country']))
df_new.head()
```

Out[54]:
```
        country                   timestamp      group landing_page  \
user_id
834778       UK  2017-01-14 23:08:43.304998    control     old_page
928468       US  2017-01-23 14:44:16.387854  treatment     new_page
822059       UK  2017-01-16 14:04:14.719771  treatment     new_page
711597       UK  2017-01-22 03:14:24.763511    control     old_page
710616       UK  2017-01-16 13:14:44.000513  treatment     new_page

         converted  intercept  ab_page  half_exp  CA  UK  US
user_id
834778           0          1        0         1   0   1   0
928468           0          1        1         1   0   0   1
822059           1          1        1         1   0   1   0
711597           0          1        0         1   0   1   0
710616           0          1        1         1   0   1   0
```

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 [55]: # Fit your model, and summarize the results
         log_mod = sm.Logit(df_new.converted,df_new[['intercept','ab_page','CA','UK']])
         results = log_mod.fit()
         print(results.summary2())

Optimization terminated successfully.
         Current function value: 0.366113
         Iterations 6
                          Results: Logit
==================================================================
Model:              Logit            No. Iterations:   6.0000
Dependent Variable: converted        Pseudo R-squared: 0.000
Date:               2021-10-01 19:21 AIC:              212781.1253
No. Observations:   290584           BIC:              212823.4439
Df Model:           3                Log-Likelihood:   -1.0639e+05
Df Residuals:       290580           LL-Null:          -1.0639e+05
Converged:          1.0000           Scale:            1.0000
-------------------------------------------------------------------
              Coef.  Std.Err.     z     P>|z|   [0.025   0.975]
-------------------------------------------------------------------
intercept    -1.9893   0.0089 -223.7628 0.0000 -2.0067 -1.9718
ab_page      -0.0149   0.0114   -1.3069 0.1912 -0.0374  0.0075
CA           -0.0408   0.0269   -1.5161 0.1295 -0.0934  0.0119
UK            0.0099   0.0133    0.7433 0.4573 -0.0162  0.0359
==================================================================
```

```
In [56]: df_new['US_ab'] = df_new.ab_page*df_new.US
         df_new['UK_ab'] = df_new.ab_page*df_new.UK
         df_new['CA_ab'] = df_new.ab_page*df_new.CA

In [57]: log_mod = sm.Logit(df_new.converted,df_new[['intercept','ab_page','CA','UK','CA_ab','UK
         results = log_mod.fit()
         print(results.summary2())

Optimization terminated successfully.
         Current function value: 0.366109
         Iterations 6
                          Results: Logit
==================================================================
Model:              Logit            No. Iterations:   6.0000
```

```
Dependent Variable: converted        Pseudo R-squared: 0.000
Date:               2021-10-01 19:28 AIC:              212782.6602
No. Observations:   290584           BIC:              212846.1381
Df Model:           5                Log-Likelihood:   -1.0639e+05
Df Residuals:       290578           LL-Null:          -1.0639e+05
Converged:          1.0000           Scale:            1.0000
-------------------------------------------------------------------
              Coef.   Std.Err.      z      P>|z|    [0.025   0.975]
-------------------------------------------------------------------
intercept    -1.9865    0.0096  -206.3440  0.0000  -2.0053  -1.9676
ab_page      -0.0206    0.0137    -1.5052  0.1323  -0.0473   0.0062
CA           -0.0175    0.0377    -0.4652  0.6418  -0.0914   0.0563
UK           -0.0057    0.0188    -0.3057  0.7598  -0.0426   0.0311
CA_ab        -0.0469    0.0538    -0.8718  0.3833  -0.1523   0.0585
UK_ab         0.0314    0.0266     1.1807  0.2377  -0.0207   0.0835
==================================================================
```

Still have too high p values, so we couldn't rely on these results.

1.1 Conclusions

Based on performed tests we don't find statistically significant results to reject the null hypothesis. So my advice to the company is to stay with the old page but also if they have enough time and money that could continue the experiment to collect more data.

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'])
```