# Analyze_ab_test_results_notebook

December 14, 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.

> **Tip**: Though it's not a mandate, students can attempt the classroom quizzes to ensure statistical numeric values are calculated correctly in many cases.

## 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]: total_rows=df.shape[0]
        print("Number of rows in the dataset",total_rows)

Number of rows in the dataset 294478
```

**c.** The number of unique users in the dataset.

```
In [4]: unique_rows=df.user_id.nunique()
        print("Number of unique users in the dataset",unique_rows)

Number of unique users in the dataset 290584
```

**d.** The proportion of users converted.

```
In [5]: propotion=(df['converted'].sum())/unique_rows
        print("The proportion of users converted",propotion)

The proportion of users converted 0.121262698566
```

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

```
In [6]: treat_old =df.query('group=="treatment" and landing_page=="old_page"').shape[0]
        control_new =df.query('group=="control" and landing_page=="new_page"').shape[0]
        total=treat_old+control_new
        print("The number of times the new_page and treatment don't line up",total)

The number of times the new_page and treatment don't line up 3893
```

**f.** Do any of the rows have missing values?

```
In [7]: #df.isnull().sum() Checking Total no of missing values
        df[df.isnull().any(axis=1)]

Out[7]: Empty DataFrame
        Columns: [user_id, timestamp, group, landing_page, converted]
        Index: []
```

### 1.0.2 ToDo 1.2

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

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

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

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

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

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

```
In [8]: # Remove the inaccurate rows, and store the result in a new dataframe df2
        df2 = df.drop(df[(df.group == 'treatment') & (df.landing_page == 'old_page')].index)
        df2 = df2.drop(df2[(df2.group == 'control') & (df2.landing_page == 'new_page')].index)
        df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 290585 entries, 0 to 294477
Data columns (total 5 columns):
user_id         290585 non-null int64
timestamp       290585 non-null object
group           290585 non-null object
landing_page    290585 non-null object
converted       290585 non-null int64
dtypes: int64(2), object(3)
memory usage: 13.3+ MB
```

```
In [9]: # Double Check all of the incorrect rows were removed from df2 -
        # Output of the statement below should be 0
        df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].sha
```

```
Out[9]: 0
```

### 1.0.3 ToDo 1.3

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

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

```
In [10]: print("Number of unique user_ids are in df2",df2.user_id.nunique())
```

```
Number of unique user_ids are in df2 290584
```

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

```
In [11]: df2[df2['user_id'].duplicated()]
```

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

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

```
In [12]: df2[df2['user_id']==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
```

**d.** Remove **one** of the rows with a duplicate **user_id**, from the **df2** dataframe.

```
In [13]: # 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 = df2.drop(1899)
         # Check again if the row with a duplicate user_id is deleted or not
         df2[df2['user_id']==773192]
```

```
Out[13]:       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?

> **Tip**: The probability you'll compute represents the overall "converted" success rate in the population and you may call it $p_{population}$.

```
In [14]: # Probability of user converting
         df2.converted.mean()
```

```
Out[14]: 0.11959708724499628
```

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

```
In [15]: # Calculate probability of conversion for control group
         p_control = df2[df2['group']=='control']['converted'].mean()

         print("Probability of conversion for control group (p_control):", p_control)
```

```
Probability of conversion for control group (p_control): 0.1203863045
```

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

```
In [16]: # Calculate probability of conversion for treatment group
         p_treatment = df2[df2['group']=='treatment']['converted'].mean()

         print("Probability of conversion for treatment group (p_treatment):", p_treatment)
```

```
Probability of conversion for treatment group (p_treatment): 0.118808065515
```

> **Tip**: The probabilities you've computed in the points (b). and (c). above can also be treated as conversion rate. Calculate the actual difference (`obs_diff`) between the conversion rates for the two groups. You will need that later.

```
In [17]: # Calculate the actual difference (obs_diff) between the conversion rates for the two g
         obs_diff=p_treatment-p_control
         obs_diff
```

```
Out[17]: -0.0015782389853555567
```

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

```
In [18]: # Calculate probability of receiving the new page.
         p_new_page = df2[df2['landing_page']=='new_page'].shape[0]/df2.shape[0]

         print("Probability of receiving the new page (p_new_page):", p_new_page)
```

```
Probability of receiving the new page (p_new_page): 0.5000619442226688
```

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

> The control group (the group with the old page) has a little higher conversion rate than the treatment group (the group with the new page) based on the probability.

## Part II - A/B Test
Since a timestamp is associated with each event, you could run a hypothesis test continuously as long as you observe the events.

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

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

### 1.0.5 ToDo 2.1

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

> Recall that you just calculated that the "converted" probability (or rate) for the old page is *slightly* higher than that of the new page (ToDo 1.4.c).

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

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

Null Hypothesis $H_0 : p_{old} \quad p_{new}$ Alternative Hypothesis $H_1 : p_{old} < p_{new}$

*OR*

Null Hypothesis $H_0 : p_{old}$ - $p_{new} \quad 0$ Alternative Hypothesis $H_1 : p_{old}$ - $p_{new} < 0$

### 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 [19]: # Calculate probability of conversion for new page
         p_new = df2[df2['landing_page']=='new_page']['converted'].mean()

         print("Probability of conversion for new page (p_new):", p_new)
```

```
Probability of conversion for new page (p_new): 0.118808065515
```

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

```
In [20]: # Calculate probability of conversion for old page
         p_old = df2[df2['landing_page']=='old_page']['converted'].mean()

         print("Probability of conversion for old page (p_old):", p_old)
```

```
Probability of conversion for old page (p_old): 0.1203863045
```

```
In [21]: p_null=p_new-p_old
         print("Difference in probability of conversion for new and old page (not under H_0):",
```

```
Difference in probability of conversion for new and old page (not under H_0): -0.00157823898536
```

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

```
In [22]: n_new = df2.query("landing_page == 'new_page'").shape[0]
         n_new
```

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

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

```
In [23]: n_old = df2.query("landing_page == 'old_page'").shape[0]
         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. *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]: p_mean=df2.query('converted == 1').shape[0] / df2.shape[0]
```

```
In [25]: # Simulate a Sample for the treatment Group

         new_page_converted = np.random.choice([1,0], size=n_new, replace = True, p=[p_mean,(1-p
         new_page_converted_mean=np.array(new_page_converted).mean()
         new_page_converted_mean
```

```
Out[25]: 0.11884935654806965
```

**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([1,0], size=n_old,  replace = True, p=[p_mean,(1-

         old_page_converted_mean=np.array(old_page_converted).mean()
         old_page_converted_mean
```

```
Out[26]: 0.11955339565235348
```

**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]: -0.0007040391042838350 9
```

**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`.

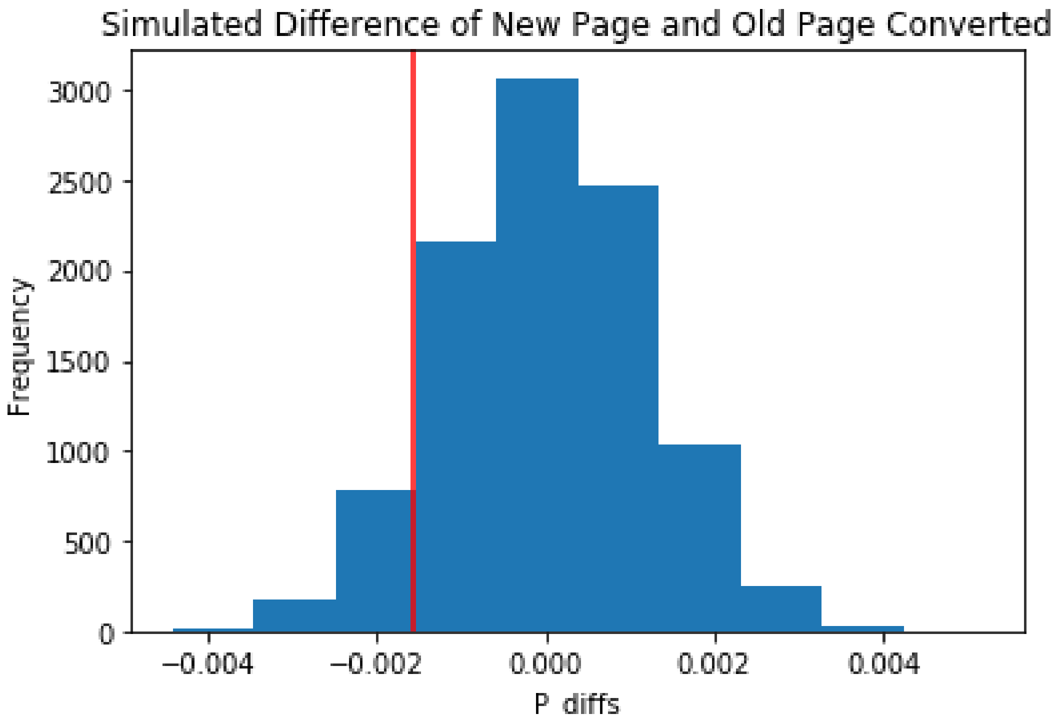
```
In [28]: # Sampling distribution
         p_diffs = []
         for i in range(10000):
             new_page_converted = np.random.choice([1, 0], size=n_new, replace = True, p=[p_mean
             old_page_converted = np.random.choice([1, 0], size=n_old, replace = True, p=[p_mean
             p_diff = new_page_converted.mean()-old_page_converted.mean()
             p_diffs.append(p_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.

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

```
In [29]: plt.hist(p_diffs);
         plt.xlabel('P_diffs');
         plt.ylabel('Frequency');
         plt.title('Simulated Difference of New Page and Old Page Converted');
         plt.axvline(x=obs_diff, color='red');
```




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

```
In [30]: p_diffs=np.array(p_diffs)
         (p_diffs > obs_diff).mean()
```

```
Out[30]: 0.9080000000000003
```

**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)".

> What we computed is the actual versus observed difference in means of converted old page and converted new page. In this scenario, our p-value surpasses the crucial threshold of 0.05, therefore we are unable to reject the null hypothesis; hence, we cannot infer that the new page converts more users than the previous page.

**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 walkthrough of the ideas that are critical to correctly thinking about statistical significance.

Fill in the statements below to calculate the: - `convert_old`: number of conversions with the old_page - `convert_new`: number of conversions with the new_page - `n_old`: number of individuals who were shown the old_page - `n_new`: number of individuals who were shown the new_page

```
In [31]: import statsmodels.api as sm

         # number of conversions with the old_page
         convert_old = df2.query('group == "control" & converted == 1').shape[0]

         # number of conversions with the new_page
         convert_new =df2.query('group == "treatment" & converted == 1').shape[0]

         # number of individuals who were shown the old_page
         n_old =  df2.shape[0] - df2.query('group == "control"').shape[0]

         # number of individuals who received new_page
         n_new = df2.query('group == "treatment"').shape[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.

> **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]: import statsmodels.api as sm

         # ToDo: Complete the sm.stats.proportions_ztest() method arguments
         z_score, p_value = sm.stats.proportions_ztest([convert_new, convert_old], [n_new, n_old
         print(z_score, p_value)
```

```
-1.28629913797 0.900830658383
```

**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 conversion rate is -1.29 standard deviations below the mean, as indicated by the z-score of -1.29. The p-value is 0.90, indicating that the null hypothesis is not rejected.

### 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?

> This is a logistic regression, since we want to know the odds of conversion, rather than a linear figure.

**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['ab_page'] = pd.get_dummies(df2['group'])['treatment']
```

**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]: df2['intercept'] = 1
         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:               2021-12-14 04:43 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
==================================================================

"""
```

In [36]:
```
np.exp(results.params)
```

Out[36]:
```
intercept    0.136863
ab_page      0.985123
dtype: float64
```

In [37]:
```
1/np.exp(results.params)
```

Out[37]:
```
intercept    7.306593
ab_page      1.015102
dtype: float64
```

**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 associated with ab_page is 0.190 here, which is similar to the previous values, but slightly higher.

**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?

> More components should be included in the model, as the existing hypotheses only include a single conversion factor, the type of page, which can show a quick relationship but does not provide insight into other elements that may influence conversion.

**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 [38]:
```
# Read the countries.csv
countries_df = pd.read_csv('./countries.csv')
```

```
In [39]: # Join with the df2 dataframe
         df_new = countries_df.set_index('user_id').join(df2.set_index('user_id'), how='inner')
```

```
In [40]: # Create the necessary dummy variables
         df_new[['CA', 'US', 'UK']] = pd.get_dummies(df_new['country'])
```

**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.

> **Tip**: Conclusions should include both statistical reasoning, and practical reasoning for the situation.
>
> **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 [41]: # Fit your model, and summarize the results
         df_new['intercept'] = 1
         logit_mod = sm.Logit(df_new['converted'], df_new[['intercept', 'ab_page', 'UK', 'CA']])
         results = logit_mod.fit()
         results.summary2()
```

```
Optimization terminated successfully.
         Current function value: 0.366113
         Iterations 6
```

```
Out[41]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                   Results: Logit
         ==================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2021-12-14 04:43 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    -1.9794    0.0127  -155.4145  0.0000  -2.0044  -1.9544
         ab_page      -0.0149    0.0114    -1.3069  0.1912  -0.0374   0.0075
         UK           -0.0099    0.0133    -0.7433  0.4573  -0.0359   0.0162
         CA           -0.0506    0.0284    -1.7835  0.0745  -0.1063   0.0050
```

```
=================================================================
"""
```

In [42]:
```
np.exp(results.params)
```

Out[42]:
```
intercept    0.138154
ab_page      0.985168
UK           0.990165
CA           0.950621
dtype: float64
```

In [43]:
```
1/np.exp(results.params)
```

Out[43]:
```
intercept    7.238314
ab_page      1.015056
UK           1.009932
CA           1.051944
dtype: float64
```

The p-value for the dummy variable ab_page does not depend on additional country dummy variables. The p-values for country dummy variables vary depending on what country is choosen as default without dummy. However, all of the independend dummy variables are not statistically significant on a 95% confidence level and we should not reject the null hypothesis.

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

```
Out[44]: 0
```