

# 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:

- [Introduction](#)
- [Part I - Probability](#)
- [Part II - A/B Test](#)
- [Part III - Regression](#)
- [Final Check](#)
- [Submission](#)

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](#) (<https://review.udacity.com/#!/rubrics/1214/view>) 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 [73]: 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 s
et up
random.seed(42)
```

## 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
<code>user_id</code>	Unique ID	Int64 values
<code>timestamp</code>	Time stamp when the user visited the webpage	-
<code>group</code>	In the current A/B experiment, the users are categorized into two broad groups. The <code>control</code> group users are expected to be served with <code>old_page</code> ; and <code>treatment</code> group users are matched with the <code>new_page</code> . However, <b>some inaccurate rows</b> are present in the initial data, such as a <code>control</code> group user is matched with a <code>new_page</code> .	['control', 'treatment']
<code>landing_page</code>	It denotes whether the user visited the old or new webpage.	['old_page', 'new_page']
<code>converted</code>	It denotes whether the user decided to pay for the company's product. Here, <code>1</code> 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 [127]: 

```
#Read in the dataset and take a look at the top few rows
df = pd.read_csv('ab_data.csv')
df.head(10)
```

Out[127]:

	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
5	936923	2017-01-10 15:20:49.083499	control	old_page	0
6	679687	2017-01-19 03:26:46.940749	treatment	new_page	1
7	719014	2017-01-17 01:48:29.539573	control	old_page	0
8	817355	2017-01-04 17:58:08.979471	treatment	new_page	1
9	839785	2017-01-15 18:11:06.610965	treatment	new_page	1

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

```
In [128]: #find the number of rows in the dataset  
len(df.index)
```

Out[128]: 294478

c. The number of unique users in the dataset.

```
In [76]: len(df.user_id.unique())
```

Out[76]: 290584

d. The proportion of users converted.

```
In [77]: conv_user = len(df.query('converted ==True'))  
print(conv_user)
```

35237

```
In [78]: proportion = (conv_user / len(df.index))*100  
print(proportion)
```

11.96591935560551

e. The number of times when the "group" is treatment but "landing\_page" is not a new\_page .

```
In [79]: treat = df.query('group == "treatment"')  
treat_old = treat.query('landing_page == "old_page"')  
len(treat_old)
```

Out[79]: 1965

```
In [80]: cont = df.query('group == "control"')  
cont_new = cont.query('landing_page == "new_page"')  
len(cont_new)
```

Out[80]: 1928

```
In [81]: treat_not_new = treat_old+cont_new  
len(treat_not_new)
```

Out[81]: 3893

f. Do any of the rows have missing values?

```
In [82]: missing_value = df[df.isnull().any(axis=1)]  
len(missing_value)
```

Out[82]: 0

## 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 webpage.

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 [83]: # Remove the inaccurate rows, and store the result in a new dataframe df2
df2 = df.copy()
df2.drop(treat_not_new.index, inplace=True)
```

```
In [84]: # 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].shape[0]
```

```
Out[84]: 0
```

## ToDo 1.3

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

a. How many unique **user\_ids** are in **df2**?

```
In [85]: len(df2.user_id.unique())
```

```
Out[85]: 290584
```

b. There is one **user\_id** repeated in **df2**. What is it?

```
In [86]: df2_duplicat = df2['user_id'].duplicated()
```

```
In [87]: df2[(df2_duplicat == True)][ "user_id"]
```

```
Out[87]: 2893    773192
         Name: user_id, dtype: int64
```

c. Display the rows for the duplicate **user\_id**?

```
In [88]: df2[(df2_duplicat == True)]
```

```
Out[88]:
```

	user_id	timestamp	group	landing_page	converted
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 [89]: # 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 duplicate user_id are not entirely identical.
         df2.drop(2893, inplace=True)
         # Check again if the row with a duplicate user_id is deleted or not
```

```
In [90]: df2_duplicat = df2['user_id'].duplicated()
         sum(df2[(df2_duplicat == True)][ "user_id"])
```

```
Out[90]: 0
```

## 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  $p_{\text{population}}$ .

```
In [91]: conv_user = len(df2.query('converted ==True'))
         p_population = (conv_user / len(df2.index))
         print(p_population)
```

```
0.11959708724499628
```

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

```
In [92]: cont_converted = df2.query('group == "control"')
```

```
In [93]: prop_convr = cont_converted.query('converted ==True')
ctr_convr_probability = len(prop_convr)/len(cont_converted)
print(ctr_convr_probability)

0.1203863045004612
```

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

```
In [94]: treat_converted = df2.query('group == "treatment"')

In [95]: prop_trt_convr = treat_converted.query('converted ==True')
trt_convr_probability = len(prop_trt_convr)/len(treat_converted)
print(trt_convr_probability)

0.11880806551510564
```

**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 [96]: # Calculate the actual difference (obs_diff) between the conversion rates for
         the two groups.
obs_diff = trt_convr_probability - ctr_convr_probability
obs_diff
```

Out[96]: -0.0015782389853555567

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

```
In [97]: new_page_prob = len(df2.query('landing_page == "new_page"))
new_page_probability = (new_page_prob / len(df2.index))
print(new_page_probability)

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.

**as conversion probability for the users in treatment group "0.1188" is less than conversion probability for the users in control group "0.1203", so there is no evidence for leading more conversion with new treatment.**

## 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.

### 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.

$H_0$   $p_{old}$  equal or greater than  $p_{new}$

$H_1$   $p_{old}$  less than  $p_{new}$

## 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 [98]: new_conv = len(df2.query('converted==True'))
```

```
In [99]: P_new = new_conv / len(df2)
print(P_new)

0.11959708724499628
```

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

```
In [100]: old_conv = len(df2.query('converted==True'))
```

```
In [101]: P_old = old_conv / len(df2)
print(P_old)

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 [102]: n_new = len(df2.query('group=="treatment"'))
n_new
```

```
Out[102]: 145310
```

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

```
In [103]: n_old = len(df2.query('group=="control"'))
n_old
```

```
Out[103]: 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 [104]: # Simulate a Sample for the treatment Group
new_page_converted = np.random.choice([0,1], n_new, P_new)
new_page_converted
```

```
Out[104]: array([1, 1, 1, ..., 0, 1, 0])
```

### 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 [105]: # Simulate a Sample for the control Group
old_page_converted = np.random.choice([0,1], n_old, P_old)
old_page_converted
```

```
Out[105]: array([0, 0, 0, ..., 0, 1, 0])
```

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

```
In [106]: new_page_converted.mean()-old_page_converted.mean()
```

```
Out[106]: -0.00036467776983217792
```

## 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 [107]: # Sampling distribution
p_diffs = []
for _ in range(10000):
    b_samp = df.sample(150, replace=True)
    new_page_converted = np.random.choice([0,1], n_new, P_new)
    old_page_converted = np.random.choice([0,1], n_old, P_old)
    p_diffs.append(new_page_converted.mean()-old_page_converted.mean())
```

```
In [108]: # Convert to numpy array
p_diffs = np.array(p_diffs)
```

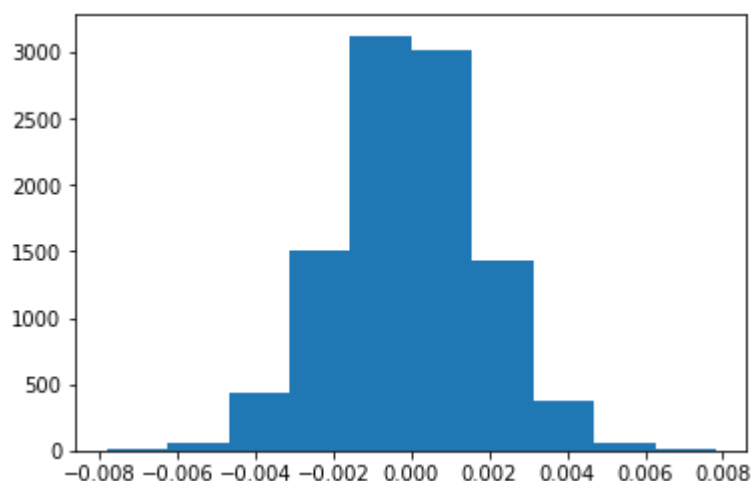
## i. Histogram

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 [109]: # Plot sampling distribution
plt.hist(p_diffs);
```

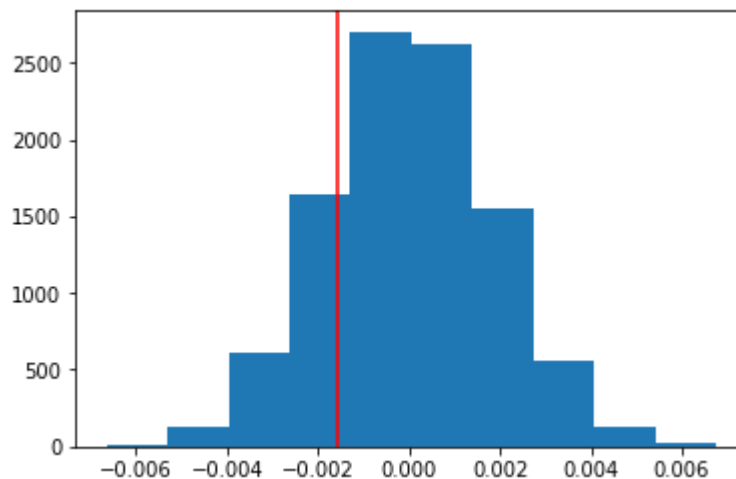


j. What proportion of the **p\_diffs** are greater than the actual difference observed in the **df2** data?

```
In [110]: # Here are 10000 draws from the sampling distribution under the null
null_vals = np.random.normal(0, p_diffs.std(), p_diffs.size)
```

```
In [111]: plt.hist(null_vals);
plt.axvline(x=obs_diff, color='red')
```

```
Out[111]: <matplotlib.lines.Line2D at 0x7f6bb04eb2b0>
```



```
In [112]: #Compute proportion of the diffs are greater than the actual difference
p_value = (null_vals > obs_diff).mean()
p_value
```

```
Out[112]: 0.8071000000000004
```

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)".

as computed the p-value is 0.7996 greater than 0.05 , which is the probability that we will observe this statistic, given the null hypothesis is true. So we fail to reject the null hypothesis. there is no a difference between the new and old pages

## I. 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 [113]: import statsmodels.api as sm

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

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

# number of individuals who were shown the old_page
n_old = len(df2.query('landing_page == "old_page"'))

# number of individuals who received new_page
n_new = len(df2.query('landing_page == "new_page"'))

convert_old, convert_new, n_old, n_new
```

```
Out[113]: (17489, 17264, 145274, 145310)
```

m. Now use `sm.stats.proportions_ztest()` to compute your test statistic and p-value. [Here](https://www.statsmodels.org/stable/generated/statsmodels.stats.proportion.proportions_ztest.html) ([https://www.statsmodels.org/stable/generated/statsmodels.stats.proportion.proportions\\_ztest.html](https://www.statsmodels.org/stable/generated/statsmodels.stats.proportion.proportions_ztest.html)) 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`.

## About the two-sample z-test

Recall that you have plotted a distribution `p_diffs` representing the difference in the "converted" probability  $(p'_{new} - p'_{old})$  for your two simulated samples 10,000 times.

Another way for comparing the mean of two independent and normal distribution is a **two-sample z-test**. You can perform the Z-test to calculate the `Z_score`, as shown in the equation below:

$$Z_{score} = \frac{(p'_{new} - p'_{old}) - (p_{new} - p_{old})}{\sqrt{\frac{\sigma_{new}^2}{n_{new}} + \frac{\sigma_{old}^2}{n_{old}}}}$$

where,

- $p'$  is the "converted" success rate in the sample
- $p_{new}$  and  $p_{old}$  are the "converted" success rate for the two groups in the population.
- $\sigma_{new}$  and  $\sigma_{old}$  are the standard deviation for the two groups in the population.
- $n_{new}$  and  $n_{old}$  represent the size of the two groups or samples (it's same in our case)

Z-test is performed when the sample size is large, and the population variance is known. The z-score represents the distance between the two "converted" success rates in terms of the standard error.

Next step is to make a decision to reject or fail to reject the null hypothesis based on comparing these two values:

- $Z_{score}$
- $Z_{\alpha}$  or  $Z_{0.05}$ , also known as critical value at 95% confidence interval.  $Z_{0.05}$  is 1.645 for one-tailed tests, and 1.960 for two-tailed test. You can determine the  $Z_{\alpha}$  from the z-table manually.

Decide if your hypothesis is either a two-tailed, left-tailed, or right-tailed test. Accordingly, reject OR fail to reject the null based on the comparison between  $Z_{score}$  and  $Z_{\alpha}$ .

Hint:

For a right-tailed test, reject null if  $Z_{score} > Z_{\alpha}$ .

For a left-tailed test, reject null if  $Z_{score} < Z_{\alpha}$ .

In other words, we determine whether or not the  $Z_{score}$  lies in the "rejection region" in the distribution. A "rejection region" is an interval where the null hypothesis is rejected iff the  $Z_{score}$  lies in that region.

Reference:

- Example 9.1.2 on this [page](https://stats.libretexts.org/Bookshelves/Introductory_Statistics/Book%3A_Introductory_Statistics_(Shafer_and_Sample_Problems/9.01%3A_Comparison_of_Two_Population_Means-_Large_Independent_Samples))  
([https://stats.libretexts.org/Bookshelves/Introductory\\_Statistics/Book%3A\\_Introductory\\_Statistics\\_\(Shafer\\_and\\_Sample\\_Problems/9.01%3A\\_Comparison\\_of\\_Two\\_Population\\_Means-\\_Large\\_Independent\\_Samples\)](https://stats.libretexts.org/Bookshelves/Introductory_Statistics/Book%3A_Introductory_Statistics_(Shafer_and_Sample_Problems/9.01%3A_Comparison_of_Two_Population_Means-_Large_Independent_Samples))), courtesy [www.stats.libretexts.org](http://www.stats.libretexts.org)

**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 [114]: 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], alternative = 'larger')
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.?

**Tip:** Notice whether the p-value is similar to the one computed earlier. Accordingly, can you reject/fail to reject the null hypothesis? It is important to correctly interpret the test statistic and p-value.

as computed the p-value is 0.905 which is still greater than 0.05 , given the null hypothesis is true. So we fail to reject the null hypothesis. there is no a difference between the new and old pages, regarding to z-score computed as -1.31092419842 still inside our critical value of 1.959 so we fail to reject null hypothesis

## Part III - A regression approach

### 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 [115]: `df2.head()`

Out[115]:

	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

```
In [116]: neighborhood_dummies = pd.get_dummies(df['group'])
df2 = df2.join(neighborhood_dummies)
df2 = df2.rename(columns={'treatment': 'ab_page'})
df2 = df2.drop('control', axis=1)
df2['intercept'] = 1
df2.head()
```

Out[116]:

	user_id	timestamp	group	landing_page	converted	ab_page	intercept
0	851104	2017-01-21 22:11:48.556739	control	old_page	0	0	1
1	804228	2017-01-12 08:01:45.159739	control	old_page	0	0	1
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0	1	1
3	853541	2017-01-08 18:28:03.143765	treatment	new_page	0	1	1
4	864975	2017-01-21 01:52:26.210827	control	old_page	1	0	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 [117]: logit_mod = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])
results = logit_mod.fit()
```

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

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



```
In [118]: from scipy import stats
stats.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df)

results.summary()
```

Out[118]: Logit Regression Results

<b>Dep. Variable:</b>	converted	<b>No. Observations:</b>	290584
<b>Model:</b>	Logit	<b>Df Residuals:</b>	290582
<b>Method:</b>	MLE	<b>Df Model:</b>	1
<b>Date:</b>	Mon, 16 May 2022	<b>Pseudo R-squ.:</b>	8.077e-06
<b>Time:</b>	17:05:02	<b>Log-Likelihood:</b>	-1.0639e+05
<b>converged:</b>	True	<b>LL-Null:</b>	-1.0639e+05
		<b>LLR p-value:</b>	0.1899

	coef	std err	z	P> z	[0.025	0.975]
<b>intercept</b>	-1.9888	0.008	-246.669	0.000	-2.005	-1.973
<b>ab_page</b>	-0.0150	0.011	-1.311	0.190	-0.037	0.007

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

#### 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 0.216 which is larger than type 1 error (.05), so that the landing page is not statistically significant in predicting viewer conversion.

null and alternative hypotheses

$H_0$   $p_{old}$  equal or greater than  $p_{new}$

$H_1$   $p_{old}$  less than  $p_{new}$

null and alternative hypotheses associated with regression model

$H_0$   $p_{old}$  equal  $p_{new}$

$H_1$   $p_{old}$  not equal  $p_{new}$

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?

**Yes I think that considering other factors affect the individual converts due to including more data could be get another conclusion.**

### 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 \(https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.join.html\)](https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.join.html) 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 [119]: # Read the countries.csv
          countries = pd.read_csv('countries.csv')
```

```
In [120]: # Join with the df2 dataframe
df2 = df2.set_index('user_id').join(countries.set_index('user_id'))
df2.head()
```

Out[120]:

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

```
In [121]: # Create the necessary dummy variables
df2[['CA', 'UK', 'US']] = pd.get_dummies(df2['country'])
df2.head()
```

Out[121]:

	timestamp	group	landing_page	converted	ab_page	intercept	country	CA	U
user_id									
851104	2017-01-21 22:11:48.556739	control	old_page	0	0	1	US	0	
804228	2017-01-12 08:01:45.159739	control	old_page	0	0	1	US	0	
661590	2017-01-11 16:55:06.154213	treatment	new_page	0	1	1	US	0	
853541	2017-01-08 18:28:03.143765	treatment	new_page	0	1	1	US	0	
864975	2017-01-21 01:52:26.210827	control	old_page	1	0	1	US	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 there are 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 [122]: # Fit your model, and summarize the results
logit_mod2 = sm.Logit(df2['converted'], df2[['intercept', 'ab_page', 'US', 'UK']])
results2 = logit_mod2.fit()
results2.summary2()
```

Optimization terminated successfully.  
 Current function value: 0.366113  
 Iterations 6

```
Out[122]:
```

Model:	Logit	Pseudo R-squared:	0.000
Dependent Variable:	converted	AIC:	212781.1253
Date:	2022-05-16 17:05	BIC:	212823.4439
No. Observations:	290584	Log-Likelihood:	-1.0639e+05
Df Model:	3	LL-Null:	-1.0639e+05
Df Residuals:	290580	LLR p-value:	0.17599
Converged:	1.0000	Scale:	1.0000
No. Iterations:	6.0000		

	Coef.	Std.Err.	z	P> z	[0.025	0.975]
<b>intercept</b>	-2.0300	0.0266	-76.2488	0.0000	-2.0822	-1.9778
<b>ab_page</b>	-0.0149	0.0114	-1.3069	0.1912	-0.0374	0.0075
<b>US</b>	0.0408	0.0269	1.5161	0.1295	-0.0119	0.0934
<b>UK</b>	0.0506	0.0284	1.7835	0.0745	-0.0050	0.1063

```
In [123]: 1/np.exp(-0.0149), np.exp(0.0408), np.exp(0.0506)
```

```
Out[123]: (1.0150115583846535, 1.0416437559600236, 1.0519020483004984)
```

**we could statistically concluded that US users are 1.04 times and UK users are 1.05 times . so it have no effect on conversion rate. as P-values larger than 0.05 so it is not statistically significant.**

In [124]:

```
# Create the necessary dummy variables for looking at an interaction between p
age and country
df2['UK_ab_page'] = df2['ab_page'] * df2['UK']
df2['US_ab_page'] = df2['ab_page'] * df2['US']
df2.head()
```

Out[124]:

	timestamp	group	landing_page	converted	ab_page	intercept	country	CA	L
user_id									
851104	2017-01-21 22:11:48.556739	control	old_page	0	0	1	US	0	
804228	2017-01-12 08:01:45.159739	control	old_page	0	0	1	US	0	
661590	2017-01-11 16:55:06.154213	treatment	new_page	0	1	1	US	0	
853541	2017-01-08 18:28:03.143765	treatment	new_page	0	1	1	US	0	
864975	2017-01-21 01:52:26.210827	control	old_page	1	0	1	US	0	

```
In [125]: #Create logistic regression for the intereaction variable between new page and
country using dummy variable
logit3 = sm.Logit(df2['converted'], df2[['intercept', 'ab_page', 'UK_ab_page', 'U
S_ab_page', 'UK', 'US']])
result3 = logit3.fit()
result3.summary()
```

Optimization terminated successfully.  
 Current function value: 0.366109  
 Iterations 6

Out[125]: Logit Regression Results

<b>Dep. Variable:</b>	converted	<b>No. Observations:</b>	290584
<b>Model:</b>	Logit	<b>Df Residuals:</b>	290578
<b>Method:</b>	MLE	<b>Df Model:</b>	5
<b>Date:</b>	Mon, 16 May 2022	<b>Pseudo R-squ.:</b>	3.482e-05
<b>Time:</b>	17:05:04	<b>Log-Likelihood:</b>	-1.0639e+05
<b>converged:</b>	True	<b>LL-Null:</b>	-1.0639e+05
		<b>LLR p-value:</b>	0.1920

	coef	std err	z	P> z	[0.025	0.975]
<b>intercept</b>	-2.0040	0.036	-55.008	0.000	-2.075	-1.933
<b>ab_page</b>	-0.0674	0.052	-1.297	0.195	-0.169	0.034
<b>UK_ab_page</b>	0.0783	0.057	1.378	0.168	-0.033	0.190
<b>US_ab_page</b>	0.0469	0.054	0.872	0.383	-0.059	0.152
<b>UK</b>	0.0118	0.040	0.296	0.767	-0.066	0.090
<b>US</b>	0.0175	0.038	0.465	0.642	-0.056	0.091

```
In [126]: np.exp(-0.0674), np.exp(0.0783), np.exp(0.0469), np.exp(0.0118), np.exp(0.0175)
)
```

Out[126]: (0.93482119806188357,  
 1.0814470441230692,  
 1.0480172021191829,  
 1.011869894648401,  
 1.0176540221507617)

By considering factors interaction could statitstically concluded that UK and new page users are 1.08 times and US and new page users are 1.048 times. as P-values larger than 0.05 so it is not statistically significant.

## 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).
1. 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.
1. 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 [1]: from subprocess import call
        call(['python', '-m', 'nbconvert', 'Analyze_ab_test_results_notebook.ipynb'])
```

```
Out[1]: 0
```

```
In [ ]:
```