

# Analyze A/B Test Results

You may either submit your notebook through the workspace here, or you may work from your local machine and submit through the next page. Either way assure that your code passes the project [RUBRIC \(https://review.udacity.com/#!/projects/37e27304-ad47-4eb0-a1ab-8c12f60e43d0/rubric\)](https://review.udacity.com/#!/projects/37e27304-ad47-4eb0-a1ab-8c12f60e43d0/rubric).

**Please save regularly.**

This project will assure you have mastered the subjects covered in the statistics lessons. The hope is to have this project be as comprehensive of these topics as possible. Good luck!

## Table of Contents

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

## Introduction

A/B tests are very commonly performed by data analysts and data scientists. It is important that you get some practice working with the difficulties of these

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 page, keep the old page, or perhaps run the experiment longer to make their decision.

**As you work through this notebook, follow along in the classroom and answer the corresponding quiz questions associated with each question.** The labels for each classroom concept are provided for each question. This will assure you are on the right track as you work through the project, and you can feel more confident in your final submission meeting the criteria. As a final check, assure you meet all the criteria on the [RUBRIC \(https://review.udacity.com/#!/projects/37e27304-ad47-4eb0-a1ab-8c12f60e43d0/rubric\)](https://review.udacity.com/#!/projects/37e27304-ad47-4eb0-a1ab-8c12f60e43d0/rubric).

## 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. Now, read in the `ab_data.csv` data. Store it in `df`. **Use your dataframe to answer the questions in Quiz 1 of the classroom.**

a. Read in the dataset 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 [24]: all_rows=df.shape[0]
print ('The number of rows in the dataset is: '+str(all_rows))
```

The number of rows in the dataset is: 290585

c. The number of unique users in the dataset.

```
In [25]: unique_users=len(df.user_id.unique())
print ('The number of unique users in the dataset is: '+str(unique_users))
```

The number of unique users in the dataset is: 290584

d. The proportion of users converted.

```
In [26]: df[df['converted']==1].count()[0]/all_rows*100
```

Out[26]: 11.959667567149026

e. The number of times the new\_page and treatment don't match.

```
In [27]: df.groupby('group').nunique()
```

Out[27]:

	user_id	timestamp	group	landing_page	converted
group					
<hr/>					
control	145274	145274	1	1	2
treatment	145310	145311	1	1	2

```
In [20]: df.groupby('landing_page').nunique()
```

Out[20]:

	user_id	timestamp	group	landing_page	converted
landing_page					
<hr/>					
new_page	145310	145311	1	1	2
old_page	145274	145274	1	1	2

```
In [30]: #old page with group treatment
OT = df.query("group == 'treatment' and landing_page == 'old_page').shape[0]
```

```
In [31]: #new page with group control  
NC = df.query("group == 'control' and landing_page == 'new_page').shape[0]
```

```
In [32]: mismatch = OT + NC  
print (str(mismatch)+' times the new_page and treatment don't match')  
  
0 times the new_page and treatment don't match
```

f. Do any of the rows have missing values?

```
In [33]: #concise summary of a dataframe  
df.info()  
#There is no missing values on dataframe  
  
<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
```

2. For the rows where **treatment** does not match with **new\_page** or **control** does not match with **old\_page**, we cannot be sure if this row truly received the new or old page. Use **Quiz 2** in the classroom to figure out how we should handle these rows.

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 [35]: #Dropping mismatch valuse
df.drop(df.query("group == 'treatment' and landing_page == 'old_page').index, inplace=True)
df.drop(df.query("group == 'control' and landing_page == 'new_page').index, inplace=True)
df.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 [37]: #export to csv after dropping mismatch values
df.to_csv('ab_data_dropped.csv', index=False)
df2 = pd.read_csv('ab_data_dropped.csv')
```

```
In [38]: # Double Check all of the correct rows were removed - this should be 0
df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].shape[0]
```

```
Out[38]: 0
```

3. Use **df2** and the cells below to answer questions for **Quiz3** in the classroom.

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

```
In [39]: unique_users=df2.user_id.nunique()
print ('Unique user_ids are in df2: '+str(unique_users))
```

```
Unique user_ids are in df2: 290584
```

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

```
In [40]: #duplicated value from user_id column
df2[df2.user_id.duplicated(keep=False)].user_id
```

```
Out[40]: 1876      773192
         2862      773192
         Name: user_id, dtype: int64
```

c. What is the row information for the repeat **user\_id**?

```
In [41]: df2[df2.user_id.duplicated(keep=False)].user_id
```

```
Out[41]:
```

	user_id	timestamp	group	landing_page	converted
<b>1876</b>	773192	2017-01-09 05:37:58.781806	treatment	new_page	0
<b>2862</b>	773192	2017-01-14 02:55:59.590927	treatment	new_page	0

d. Remove **one** of the rows with a duplicate **user\_id**, but keep your dataframe as **df2**.

```
In [42]: df2.drop_duplicates('user_id', inplace=True)
```

```
In [43]: #Double check the existence of duplicated values on dataframe df2
sum(df2.duplicated())
```

```
Out[43]: 0
```

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 [44]: all_rows=df2.shape[0]
p_ind_conv=df2[df2['converted']==1].count()[0]/all_rows
print ('The probability of an individual converting regardless of the page they receive '+str(p_ind_conv))
```

The probability of an individual converting regardless of the page they receive 0.119597087245

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

```
In [45]: df2.query("group == 'control'").converted.mean()
```

```
Out[45]: 0.1203863045004612
```

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

```
In [46]: df2.query("group == 'treatment'").converted.mean()
```

```
Out[46]: 0.11880806551510564
```

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

```
In [47]: #probability of recieving the new page  
df2.query("landing_page == 'new_page'").shape[0] / df2.landing_page.shape[0]
```

```
Out[47]: 0.5000619442226688
```

e. Consider your results from parts (a) through (d) above, and explain below whether you think there is sufficient evidence to conclude that the new treatment page leads to more conversions.

**The numbers of individuals who converted from group "treatment" and "control" are almost same that equals to 12% of each group. There is no concrete evidence that one page leads to more conversions.**

## Part II - A/B Test

Notice that because of the time stamp associated with each event, you could technically run a hypothesis test continuously as each observation was observed.

However, then the hard question is 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. 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 your null and alternative hypotheses be? You can state your hypothesis in terms of words or in terms of  $p_{old}$  and  $p_{new}$ , which are the converted rates for the old and new pages.

**Null hypothesis:**

$$H_0 : p_{old} - p_{new} \geq 0$$

**Alternative hypothesis:**

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

$p_{old}$  and  $p_{new}$  are the converted rates for the old and new pages respectively.



2. Assume under the null hypothesis,  $p_{new}$  and  $p_{old}$  both have "true" success rates equal to the **converted** success rate regardless of page - that is  $p_{new}$  and  $p_{old}$  are equal. Furthermore, assume they are equal to the **converted** rate in **ab\_data.csv** regardless of the page.

Use a sample size for each page equal to the ones in **ab\_data.csv**.

Perform the sampling distribution for the difference in **converted** between the two pages over 10,000 iterations of calculating an estimate from the null.

Use the cells below to provide the necessary parts of this simulation. If this doesn't make complete sense right now, don't worry - you are going to work through the problems below to complete this problem. 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?

```
In [48]: P_new = df2.converted.mean()  
P_new
```

```
Out[48]: 0.11959708724499628
```

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

```
In [49]: P_old = df2.converted.mean()  
P_old
```

```
Out[49]: 0.11959708724499628
```

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

```
In [50]: n_new = df2.query("group == 'treatment'").shape[0]
n_new
```

```
Out[50]: 145310
```

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

```
In [51]: n_old = df2.query("group == 'control'").shape[0]
n_old
```

```
Out[51]: 145274
```

e. Simulate  $n_{new}$  transactions with a conversion rate of  $p_{new}$  under the null. Store these  $n_{new}$  1's and 0's in **new\_page\_converted**.

```
In [52]: new_converted = np.random.choice([0, 1], size = n_new, p = [P_new, 1 - P_new])
new_converted
```

```
Out[52]: array([1, 1, 1, ..., 1, 1, 0])
```

f. Simulate  $n_{old}$  transactions with a conversion rate of  $p_{old}$  under the null. Store these  $n_{old}$  1's and 0's in **old\_page\_converted**.

```
In [53]: old_converted = np.random.choice([0, 1], size = n_old, p = [P_old, 1 - P_old])
old_converted
```

```
Out[53]: array([1, 1, 0, ..., 1, 1, 1])
```

g. Find  $p_{new} - p_{old}$  for your simulated values from part (e) and (f).

```
In [54]: differ=new_converted.mean() - old_converted.mean()
differ
```

```
Out[54]: 0.00055953242651318735
```

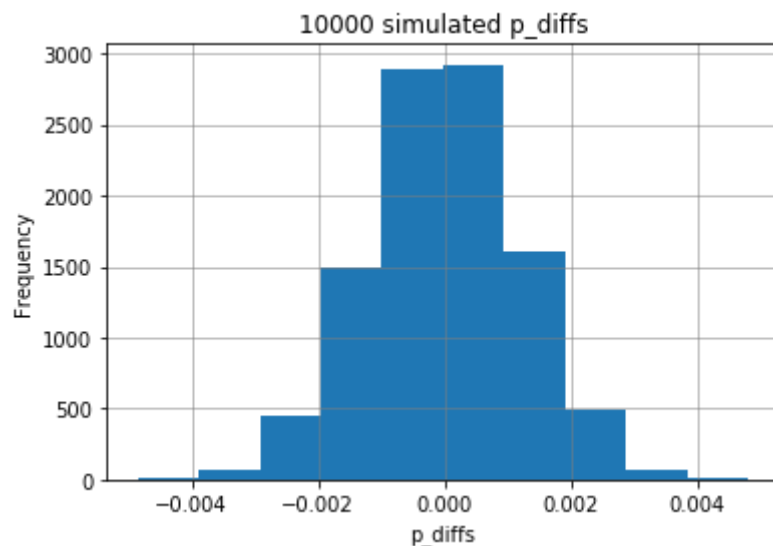
h. Create 10,000  $p_{new} - p_{old}$  values using the same simulation process you used in parts (a) through (g) above. Store all 10,000 values in a NumPy array called **p\_diffs**.

```
In [55]: diffs = []

for _ in range(10000):
    new_converted = np.random.choice([0, 1], size = n_new, p = [P_new, 1 - P_new]).mean()
    old_converted = np.random.choice([0, 1], size = n_old, p = [P_old, 1 - P_old]).mean()
    differ = new_converted - old_converted
    diffs.append(differ)
```

i. 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.

```
In [61]: plt.hist(diffs);
plt.ylabel('Frequency')
plt.xlabel('p_diffs')
plt.title('10000 simulated p_diffs');
plt.grid(color='grey',linestyle='-',linewidth=0.5)
```



j. What proportion of the **p\_diffs** are greater than the actual difference observed in **ab\_data.csv**?

```
In [62]: p_diffs = np.array(diffs)
p_diffs
```

```
Out[62]: array([ 1.30280683e-03,  5.71172632e-05, -1.05102933e-03, ...,
                -3.91551494e-05, -2.26219374e-03, -4.31711581e-04])
```

```
In [65]: actual_difference = df2.query('group == "treatment"').converted.mean() - df2.query('group == "contro
1"').converted.mean()
actual_difference
```

```
Out[65]: -0.0015782389853555567
```

```
In [66]: (p_diffs > actual_difference).mean()
```

```
Out[66]: 0.90500000000000003
```

k. Please explain using the vocabulary you've learned in this course what you just computed in part j. What is this value called in scientific studies? What does this value mean in terms of whether or not there is a difference between the new and old pages?

**In part j, we calculated the p value which is the probability of getting a more extreme value if the  $H_0$  hypothesis is true (statistic). p-value about 0.9 looks likely that  $H_0$  hypothesis is true. There is no evidence to reject the null hypothesis which consider that old pages are not worse than the new pages.**

l. 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 below to calculate the number of conversions for each page, as well as the number of individuals who received each page. Let `n_old` and `n_new` refer the the number of rows associated with the old page and new pages, respectively.

```
In [86]: import statsmodels.api as sm

convert_old = df2.query('group == "control").converted.sum()
convert_new = df2.query('group == "treatment").converted.sum()
n_old = n_old = df2.query("landing_page == 'old_page').shape[0]
n_new = n_old = df2.query("landing_page == 'new_page').shape[0]
```

m. Now use `stats.proportions_ztest` to compute your test statistic and p-value. [Here](https://docs.w3cub.com/statsmodels/generated/statsmodels.stats.proportion.proportions_ztest/) ([https://docs.w3cub.com/statsmodels/generated/statsmodels.stats.proportion.proportions\\_ztest/](https://docs.w3cub.com/statsmodels/generated/statsmodels.stats.proportion.proportions_ztest/)) is a helpful link on using the built in.

```
In [87]: z_score, p_value = sm.stats.proportions_ztest([convert_old, convert_new], [n_old, n_new], alternative
           = 'smaller')
z_score, p_value
```

```
Out[87]: (1.2862991379657529, 0.90083065838269638)
```

```
In [88]: from scipy.stats import norm

norm.cdf(z_score)
```

```
Out[88]: 0.90083065838269638
```

```
In [89]: norm.ppf(1-(0.05/2)) #critical value at 95% confidence interval
```

```
Out[89]: 1.959963984540054
```

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

**The z score that equals to 1.29 and p value 0.9 doesn't exceed the critical value 1.96 at 95% confidence interval. So we can't reject the  $H_0$  hypothesis. Also agree with the findings in parts j. and k**

## Part III - A regression approach

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 is either a conversion or no conversion, what type of regression should you be performing in this case?

**We are working with rows with 0 and 1 values. That's why we are use Logistic Regression**

b. The goal is to use **statsmodels** to fit the regression model you specified in part **a.** to see if there is a significant difference in conversion based on which page a customer receives. However, you first need to create in **df2** a column for the intercept, and create a dummy variable column for which page each user received. Add an **intercept** column, as well as an **ab\_page** column, which is 1 when an individual receives the **treatment** and 0 if **control**.

```
In [90]: df2['intercept'] = 1
df2[['ab_page2', 'ab_page']] = pd.get_dummies(df2['group'])
```

```
In [91]: df2 = df2.drop('ab_page2', axis = 1)
```

```
In [92]: df2.head()
```

Out[92]:

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

c. Use **statsmodels** to instantiate your regression model on the two columns you created in part b., then fit the model using the two columns you created in part **b.** to predict whether or not an individual converts.

```
In [93]: log_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 [94]: results = log_mod.fit()
```

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

```
In [110]: import statsmodels.api as sm
import scipy.stats as stats
stats.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df) #I've got an error and used this
# https://stackoverflow.com/questions/49814258/statsmodel-attributeerror-module-scipy-stats-has-no-at
tribute-chisqprob

results.summary()
```

Out[110]: 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>	Thu, 14 May 2020	<b>Pseudo R-squ.:</b>	8.077e-06
<b>Time:</b>	15:55:17	<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**?

**Hint:** 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**?

the p-value associated with **ab\_page** is 0.19 that lower from the result 0.9 that taken at Part2. Because the null and alternative hypotheses are different on each Parts. For this part:

$$H_0 : p_{old} = p_{new}$$

$$H_1 : p_{old} \neq p_{new}$$



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?

Adding more values to a regression model allows us to define the relation of a response variable to an explanatory variable, while holding other variables constant. May be used to identify outliers. But may produce inaccurate results due to correlated errors.

g. Now along with testing if the conversion rate changes for different pages, also add an effect based on which country a user lives in. You will need to read in the **countries.csv** dataset and merge together your datasets on the appropriate rows. [Here \(https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.join.html\)](https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.join.html) are the docs for joining tables.

Does it appear that country had an impact on conversion? Don't forget to create dummy variables for these country columns - **Hint: You will need two columns for the three dummy variables.** Provide the statistical output as well as a written response to answer this question.

```
In [117]: df_countries = pd.read_csv('countries.csv')
df3 = df_countries.set_index('user_id').join(df2.set_index('user_id'), how='inner')
df3.head()
```

Out[117]:

	country	timestamp	group	landing_page	converted	intercept	ab_page
user_id							
834778	UK	2017-01-14 23:08:43.304998	control	old_page	0	1	0
928468	US	2017-01-23 14:44:16.387854	treatment	new_page	0	1	1
822059	UK	2017-01-16 14:04:14.719771	treatment	new_page	1	1	1
711597	UK	2017-01-22 03:14:24.763511	control	old_page	0	1	0
710616	UK	2017-01-16 13:14:44.000513	treatment	new_page	0	1	1

```
In [118]: df3.country.unique()
```

Out[118]: array(['UK', 'US', 'CA'], dtype=object)

```
In [119]: df3[['UK', 'US']] = pd.get_dummies(df3['country'])[['UK', 'US']]
df3.head()
```

Out[119]:

	country	timestamp	group	landing_page	converted	intercept	ab_page	UK	US
user_id									
834778	UK	2017-01-14 23:08:43.304998	control	old_page	0	1	0	1	0
928468	US	2017-01-23 14:44:16.387854	treatment	new_page	0	1	1	0	1
822059	UK	2017-01-16 14:04:14.719771	treatment	new_page	1	1	1	1	0
711597	UK	2017-01-22 03:14:24.763511	control	old_page	0	1	0	1	0
710616	UK	2017-01-16 13:14:44.000513	treatment	new_page	0	1	1	1	0

```
In [120]: log_mod = sm.Logit(df3['converted'], df3[['intercept', 'UK', 'US']])#statistical output
```

```
In [121]: results = log_mod.fit()
```

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

```
In [122]: results.summary()
```

Out[122]: Logit Regression Results

<b>Dep. Variable:</b>	converted	<b>No. Observations:</b>	290584
<b>Model:</b>	Logit	<b>Df Residuals:</b>	290581
<b>Method:</b>	MLE	<b>Df Model:</b>	2
<b>Date:</b>	Thu, 14 May 2020	<b>Pseudo R-squ.:</b>	1.521e-05
<b>Time:</b>	16:00:03	<b>Log-Likelihood:</b>	-1.0639e+05
<b>converged:</b>	True	<b>LL-Null:</b>	-1.0639e+05
		<b>LLR p-value:</b>	0.1984

	coef	std err	z	P> z	[0.025	0.975]
<b>intercept</b>	-2.0375	0.026	-78.364	0.000	-2.088	-1.987
<b>UK</b>	0.0507	0.028	1.786	0.074	-0.005	0.106
<b>US</b>	0.0408	0.027	1.518	0.129	-0.012	0.093

h. 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 significant effects on conversion. Create the necessary additional columns, and fit the new model.

Provide the summary results, and your conclusions based on the results.

```
In [126]: log_mod = sm.Logit(df3['converted'], df3[['intercept', 'UK', 'US', 'ab_page']])
```

```
In [127]: results = log_mod.fit()
```

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

In [128]: `results.summary()`

Out[128]: Logit Regression Results

<b>Dep. Variable:</b>	converted	<b>No. Observations:</b>	290584
<b>Model:</b>	Logit	<b>Df Residuals:</b>	290580
<b>Method:</b>	MLE	<b>Df Model:</b>	3
<b>Date:</b>	Thu, 14 May 2020	<b>Pseudo R-squ.:</b>	2.323e-05
<b>Time:</b>	16:00:14	<b>Log-Likelihood:</b>	-1.0639e+05
<b>converged:</b>	True	<b>LL-Null:</b>	-1.0639e+05
		<b>LLR p-value:</b>	0.1760

	coef	std err	z	P> z	[0.025	0.975]
<b>intercept</b>	-2.0300	0.027	-76.249	0.000	-2.082	-1.978
<b>UK</b>	0.0506	0.028	1.784	0.074	-0.005	0.106
<b>US</b>	0.0408	0.027	1.516	0.130	-0.012	0.093
<b>ab_page</b>	-0.0149	0.011	-1.307	0.191	-0.037	0.007

## Finishing Up

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 report to make sure that it satisfies all the areas of the rubric (found on the project submission page at the end of the lesson). You should also probably remove all of the "Tips" like this one so that the presentation is as polished as possible.

## Directions to Submit

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).

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.

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

```
Out[129]: 0
```