

# 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)

from IPython.display import display
```

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')
display(df.head())
```

	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]: display(df.shape)
display(df.shape[0])
```

```
(294478, 5)
```

```
294478
```

c. The number of unique users in the dataset.

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

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

d. The proportion of users converted.

```
In [5]: display(df.converted.mean())
df.converted.mean()*100
```

```
0.11965919355605512
```

```
Out[5]: 11.965919355605511
```

e. The number of times the `new_page` and `treatment` don't match.

```
In [6]: # we can use query() so as to filter the combination of groups and landing page which are feasible.

ans_1e = df.query('group=="control" and landing_page=="new_page" or group=="treatment" and landing_page=="old_page").user_id.count()

ans_1e
```

Out[6]: 3893

f. Do any of the rows have missing values?

```
In [7]: df.isnull().sum()
```

```
Out[7]: user_id      0
        timestamp    0
        group        0
        landing_page  0
        converted     0
        dtype: int64
```

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 [8]: # we can slice data for the combination that we DONT want, now get their indices and store them in a variable to drop
        # it in when needed.

new_pg = df[(df.group == 'control') & (df.landing_page == 'new_page')]
old_pg = df[(df.group == 'treatment') & (df.landing_page == 'old_page')]

idx = new_pg.append(old_pg).index
idx
```

```
Out[8]: Int64Index([    22,    240,    490,    846,    850,    988,   1198,    1
354,
                  1474,   1877,
                  ...
                293240, 293302, 293391, 293443, 293530, 293773, 293817, 293
917,
                  294014, 294252],
                dtype='int64', length=3893)
```

In [9]: *# We create a new dataframe df2 which will have all the not required indexed rows dropped.*

```
df2 = df.drop(idxs)
```

```
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 [10]: *# 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[10]: 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 [11]: df2.nunique()['user\_id']

Out[11]: 290584

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

In [12]: df2[df2.user\_id.duplicated()]

Out[12]:

	user_id	timestamp	group	landing_page	converted
2893	773192	2017-01-14 02:55:59.590927	treatment	new_page	0

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

```
In [13]: df2[df2.user_id == 773192]
```

```
Out[13]:
```

	user_id	timestamp	group	landing_page	converted
1899	773192	2017-01-09 05:37:58.781806	treatment	new_page	0
2893	773192	2017-01-14 02:55:59.590927	treatment	new_page	0

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

```
In [14]: # Let's remove the row with index = 2893
```

```
df2.drop([2893], axis=0, inplace=True)
```

```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 290584 entries, 0 to 294477
Data columns (total 5 columns):
user_id      290584 non-null int64
timestamp    290584 non-null object
group        290584 non-null object
landing_page 290584 non-null object
converted    290584 non-null int64
dtypes: int64(2), object(3)
memory usage: 13.3+ MB
```

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]: # Here we can simply consider taking mean of converted column since the
         # page given to them doesn't matter
```

```
ans_4a = df2.converted.mean()
```

```
ans_4a
```

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

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

```
In [16]: ans_4b = df2[df2['group']=='control'].converted.mean()
```

```
ans_4b
```

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

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

```
In [17]: ans_4c = df2[df2['group']=='treatment'].converted.mean()  
ans_4c
```

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

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

```
In [18]: ans_4d = (df2['landing_page'] == 'new_page').mean()  
ans_4d
```

```
Out[18]: 0.50006194422266881
```

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.

**Your answer goes here.**

- The probability results show us that the 'converted' churn for 'control' group is higher than the 'treatment' group.
- The probability for those who received the new page is 0.50006194422266881 ~ 0.5
- Due to lack of concrete evidence, one cannot say if one page leads to more conversions or not.

## 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.

$$H_0 : p_{new} \leq p_{old} , H_1 : p_{new} > p_{old}$$

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 [19]: p_new = df2['converted'].mean()  
p_new
```

```
Out[19]: 0.11959708724499628
```

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

```
In [20]: p_old = df2.converted.mean()  
p_old
```

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

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

```
In [21]: n_new = df2[df2.group == 'treatment'].shape[0]  
n_new
```

```
Out[21]: 145310
```

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

```
In [22]: n_old = df2[df2.group == 'control'].shape[0]  
n_old
```

```
Out[22]: 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 [23]: new_page_converted = np.random.binomial(n_new, p_new)
```

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 [24]: old_page_converted = np.random.binomial(n_old, p_old)
```

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

```
In [25]: new_page_converted/n_new - old_page_converted/n_old
```

```
Out[25]: 0.0006928957305575717
```

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**.

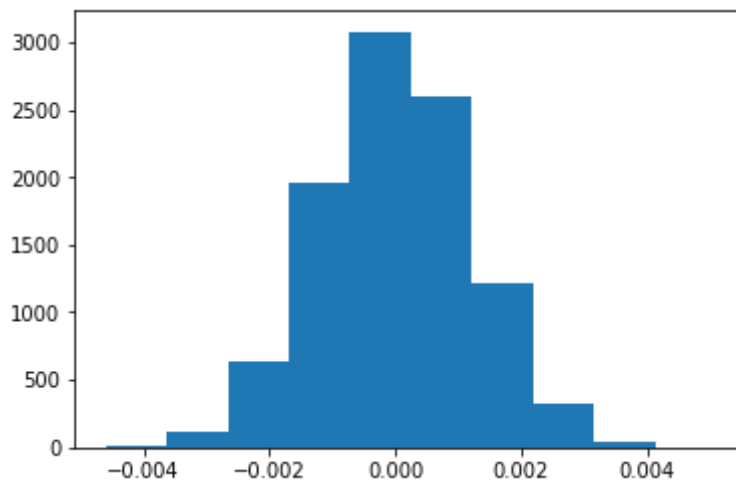
```
In [26]: p_diffs=[]

for _ in range(10000):
    new_page_converted = np.random.binomial(n_new, p_new)
    old_page_converted = np.random.binomial(n_old, p_old)
    diff = new_page_converted/n_new - old_page_converted/n_old
    p_diffs.append(diff)
```

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 [27]: plt.hist(p_diffs);
```



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

```
In [28]: # first we will calculate the actual difference

act_dif = df2[df2['group']=='treatment'].converted.mean() - df2[df2['group']=='control'].converted.mean()
display(act_dif)

# now we calculate the proportion of p_diffs that are greater than the actual difference
act_dif

p_diffs = np.array(p_diffs)
(act_dif < p_diffs).mean()

-0.0015782389853555567
```

```
Out[28]: 0.90759999999999996
```

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?

**Put your answer here.**

- The value we calculated in part-j is called as p-value.
  - The p-value or probability value is the probability of obtaining test results at least as extreme as the results actually observed during the test, assuming that the null hypothesis is true.
- A large / very large p-value gives us indication that the evidence against null hypothesis is weak. And hence, we fail to reject the null hypothesis, which means that the old page's conversion rate is higher than the new page.

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 to the number of rows associated with the old page and new pages, respectively.

```
In [29]: import statsmodels.api as sm

convert_old = df2.query('landing_page == "old_page" and converted == 1')
            .shape[0]
convert_new = df2.query('landing_page == "new_page" and converted == 1')
            .shape[0]
n_old = df2[df2.group == 'control'].shape[0]
n_new = df2[df2.group == 'treatment'].shape[0]

display(n_old)
display(n_new)
```

/opt/conda/lib/python3.6/site-packages/statsmodels/compat/pandas.py:56:  
FutureWarning: The pandas.core.datetools module is deprecated and will  
be removed in a future version. Please use the pandas.tseries module in  
stead.  
 from pandas.core import datetools

145274

145310

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

print(z_score, p_value)

1.31092419842 0.905058312759
```

```
In [ ]:
```

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.?

```
In [31]: from scipy.stats import norm

# how significant is z_score
print(norm.cdf(z_score))

# for one-sided test, assumed at 95% CI
print(norm.ppf(1-(0.05)))

0.905058312759
1.64485362695
```

**Put your answer here.**

- The z\_score (i.e. 1.31) is less than the critical value of 1.64485.
- Hence, we fail to reject Null Hypothesis which suggests that Old page conversion is higher than the New page.
- Yes, I agree with the findings in the part j & 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?

**Put your answer here.**

- Logistic Regression, since each row is either a conversion or no conversion (category) type we can clearly see its a logistic regression problem.

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 [32]: #. We will add an intercept column, as well as an ab_page column,
# which is 1 when an individual receives the treatment and 0 if control.

df2['intercept'] = 1
df2[['ab_page2', 'ab_page']] = pd.get_dummies(df2['group'])
df2[['control', 'treatment']] = pd.get_dummies(df2['group'])
df2 = df2.drop('ab_page2', axis = 1)
df2.head()
```

Out[32]:

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

In [ ]:

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 [33]: import statsmodels.api as sm

log_reg = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])
results = log_reg.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 [34]: results.summary2()
```

```
Out[34]:
```

Model:	Logit	No. Iterations:	6.0000
Dependent Variable:	converted	Pseudo R-squared:	0.000
Date:	2020-04-18 02:55	AIC:	212780.3502
No. Observations:	290584	BIC:	212801.5095
Df Model:	1	Log-Likelihood:	-1.0639e+05
Df Residuals:	290582	LL-Null:	-1.0639e+05
Converged:	1.0000	Scale:	1.0000

	Coef.	Std.Err.	z	P> z	[0.025	0.975]
intercept	-1.9888	0.0081	-246.6690	0.0000	-2.0046	-1.9730
ab_page	-0.0150	0.0114	-1.3109	0.1899	-0.0374	0.0074

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

**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**?

**Put your answer here.**

- The p\_value associated with ab\_page is 0.1899 ~ 0.190
  - This one was a two-sided test, in Part-II it was one-sided test. We will test for the not equal type hypothesis.

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

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?

**Put your answer here.**

- Other attributes like age or gender etc to the model can be unfruitful for our model.
- The effects of Simpson's paradox and similar phenomenon can come to play hence wont be so much useful.
- We lose some degree of freedom when we add a new predictor variable.
- There is a possibility of adding highly correlated predictors which can bring in Multi-collinearity leading to unreliable and unstable results from our regression model.

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 [35]: countries = pd.read_csv('countries.csv')
df_new = countries.set_index('user_id').join(df2.set_index('user_id'), how='inner')
df_new.head()
```

Out[35]:

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

```
In [37]: # Creating dummy variables
df_new['intercept'] = 1
df_new[['UK', 'US']] = pd.get_dummies(df_new['country'])[['UK', 'US']]
```

```
In [38]: df_new['US_ab_page'] = df_new['US'] * df_new['ab_page']
df_new['UK_ab_page'] = df_new['UK'] * df_new['ab_page']
```

In [39]: `df_new.head()`

Out[39]:

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

```
In [41]: lm_mod = sm.Logit(df_new['converted'], df_new[['intercept', 'US', 'UK']])

results = lm_mod.fit()

results.summary2()
```

Optimization terminated successfully.  
Current function value: 0.366116  
Iterations 6

Out[41]:

Model:	Logit	No. Iterations:	6.0000
Dependent Variable:	converted	Pseudo R-squared:	0.000
Date:	2020-04-18 03:00	AIC:	212780.8333
No. Observations:	290584	BIC:	212812.5723
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	-2.0375	0.0260	-78.3639	0.0000	-2.0885	-1.9866
US	0.0408	0.0269	1.5178	0.1291	-0.0119	0.0935
UK	0.0507	0.0284	1.7863	0.0740	-0.0049	0.1064

The Statistical analysis shows that the p-value for both the country attributes is larger than 0.05; hence indicating that there is no statistical evidence about it having a significant impact on the conversion.

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 [42]: # I have tried and included the ab_page column from part II exercise
mod_2 = sm.Logit(df_new['converted'], df_new[['intercept', 'UK', 'US',
'ab_page', 'US_ab_page', 'UK_ab_page']])
```

```
In [43]: results= mod_2.fit()

results.summary2()
```

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

```
Out[43]:
```

Model:	Logit	No. Iterations:	6.0000
Dependent Variable:	converted	Pseudo R-squared:	0.000
Date:	2020-04-18 03:00	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]
<b>intercept</b>	-2.0040	0.0364	-55.0077	0.0000	-2.0754	-1.9326
<b>UK</b>	0.0118	0.0398	0.2957	0.7674	-0.0663	0.0899
<b>US</b>	0.0175	0.0377	0.4652	0.6418	-0.0563	0.0914
<b>ab_page</b>	-0.0674	0.0520	-1.2967	0.1947	-0.1694	0.0345
<b>US_ab_page</b>	0.0469	0.0538	0.8718	0.3833	-0.0585	0.1523
<b>UK_ab_page</b>	0.0783	0.0568	1.3783	0.1681	-0.0330	0.1896

The output above shows that even after the including the column (which defines interaction between page and country) also there is no statistical evident to indicate that there is an impact on the conversion since p-values are all greater than 0.05.



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