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). **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

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).

Part I - Probability

To get started, let's import our libraries.

In [2]:

```
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)
```

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 [3]:

```
df=pd.read_csv('ab_data.csv')
df.head()
```

Out[3]:

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

In [4]:

```
df.shape[0]
```

Out[4]:

```
294478
```

c. The number of unique users in the dataset.

In [5]:

```
df['user_id'].nunique()
```

Out[5]:

```
290584
```

d. The proportion of users converted.

In [6]:

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

Out[6]:

```
0.12126269856564711
```

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

In [12]:

```
control_newpage=df.query('group=="control" & landing_page=="new_page"')['user_
id'].count() # the control group received new page
treatment_oldpage=df.query('group=="treatment" & landing_page=="old_page"')['u
ser_id'].count() # the treatment group received old page
not_line_up=control_newpage+treatment_oldpage
not_line_up
```

Out[12]:

```
3893
```

f. Do any of the rows have missing values?

In [13]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 294478 entries, 0 to 294477
Data columns (total 5 columns):
user_id         294478 non-null int64
timestamp       294478 non-null object
group           294478 non-null object
landing_page    294478 non-null object
converted       294478 non-null int64
dtypes: int64(2), object(3)
memory usage: 11.2+ MB
```

No. The data has all values.

2. For the rows where **treatment** is not aligned with **new_page** or **control** is not aligned with **old_page**, we cannot be sure if this row truly received the new or old page. Use **Quiz 2** in the classroom to provide 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 [24]:

```
treatment_new=df.query('group=="treatment" & landing_page=="new_page"')
control_old=df.query('group=="control" & landing_page=="old_page"')
df2=treatment_new.merge(control_old, how='outer')
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 290585 entries, 0 to 290584
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 [25]:

```
# 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[25]:

```
0
```

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

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

In [26]:

```
df2.nunique()
```

Out[26]:

```
user_id         290584
timestamp       290585
group                2
landing_page         2
converted            2
dtype: int64
```

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

In [27]:

```
df2[df2.duplicated(['user_id'])]
```

Out[27]:

	user_id	timestamp	group	landing_page	converted
1404	773192	2017-01-14 02:55:59.590927	treatment	new_page	0

c. What is the row information for the repeat **user_id**?

In [28]:

```
df2[df2.duplicated(['user_id'], keep=False)]
```

Out[28]:

	user_id	timestamp	group	landing_page	converted
938	773192	2017-01-09 05:37:58.781806	treatment	new_page	0
1404	773192	2017-01-14 02:55:59.590927	treatment	new_page	0

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

In [29]:

```
df2.drop([938],inplace=True) #remove one of the rows, each row has 0 convertio
n, which row to remove doesn't matter.
df2.info() #290584 instead of 290585
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 290584 entries, 0 to 290584
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 below cells 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 [36]:

```
probability_regardless=df2['converted'].sum()/df2.shape[0]
probability_regardless
```

Out[36]:

```
0.11959708724499628
```

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

In [37]:

```
control_probability=df2.query('group=="control"')['converted'].sum()/df2.query
('group=="control"').shape[0]
control_probability
```

Out[37]:

```
0.1203863045004612
```

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

In [38]:

```
experiment_probability=df2.query('group=="treatment"')['converted'].sum()/df2.
query('group=="treatment"').shape[0]
experiment_probability
```

Out[38]:

```
0.11880806551510564
```

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

In [39]:

```
new_page_received=df2.query('landing_page=="new_page"').shape[0]/df2.shape[0]
new_page_received
```

Out[39]:

```
0.5000619442226688
```

e. Use the results in the previous two portions of this question to suggest if you think there is evidence that one page leads to more conversions? Write your response below.

The control group showed rate of 0.1203 conversion while the treatment group - 0.1188. It looks like that the treatment generated less conversion than the control group. But we have to keep in mind change aversion effect and novelty effect. After running the t-test, we'll be more sure about evidence of the pages 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.

H0: $p_{old} >= p_{new}$
H1: $p_{old} < p_{new}$

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 **convert rate** for p_{new} under the null?

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

c. What is n_{new}?

In [241]:

```
n_new=df2.groupby('landing_page')['user_id'].count().loc['new_page']
n_new
```

Out[241]:

```
145310
```

d. What is n_{old} ?

In [242]:

```
n_old=df2.groupby('landing_page')['user_id'].count().loc['old_page']
n_old
```

Out[242]:

```
145274
```

e. Simulate n_{new} transactions with a convert rate of p_{new} under the null. Store these n_{new} 1's and 0's in **new_page_converted**.

In [46]:

```
sample_df2=df2.sample(df2.shape[0],replace=True)
new_page_converted=sample_df2.query('group=="treatment"')['converted'].sum()/s
ample_df2.query('group=="treatment"').shape[0]
new_page_converted
```

Out[46]:

```
0.11917187629093112
```

f. Simulate n_{old} transactions with a convert rate of p_{old} under the null. Store these n_{old} 1's and 0's in **old_page_converted**.

In [47]:

```
sample_df2=df2.sample(df2.shape[0],replace=True)
old_page_converted=sample_df2.query('group=="control"')['converted'].sum()/sam
ple_df2.query('group=="control"').shape[0]
old_page_converted
```

Out[47]:

```
0.12124963889232801
```

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

In [48]:

```
new_page_converted-old_page_converted
```

Out[48]:

```
-0.0020777626013968875
```

h. Simulate 10,000 p_{new} - p_{old} values using this same process similarly to the one you calculated in parts **a. through g.** above. Store all 10,000 values in **p_diffs**.

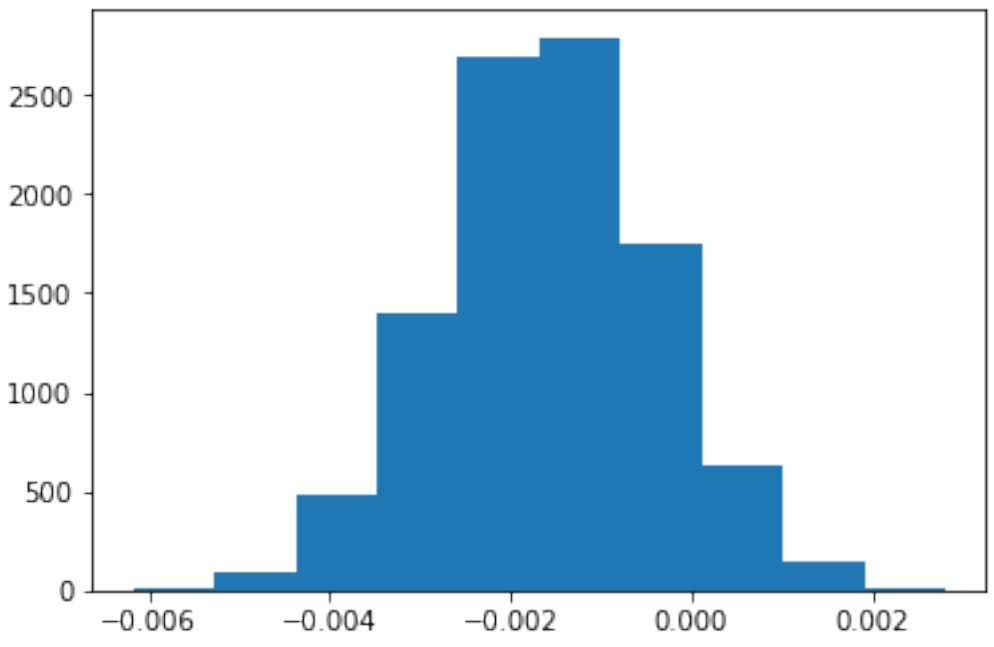
In [49]:

```
p_diffs=[]
for _ in range(10000):
    boot_sample=df2.sample(df2.shape[0],replace=True)
    new_page=boot_sample.query('group=="treatment"')['converted'].sum()/boot_s
ample.query('group=="treatment"').shape[0]
    old_page=boot_sample.query('group=="control"')['converted'].sum()/boot_sam
ple.query('group=="control"').shape[0]
    p_diffs.append(new_page-old_page)
```

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.

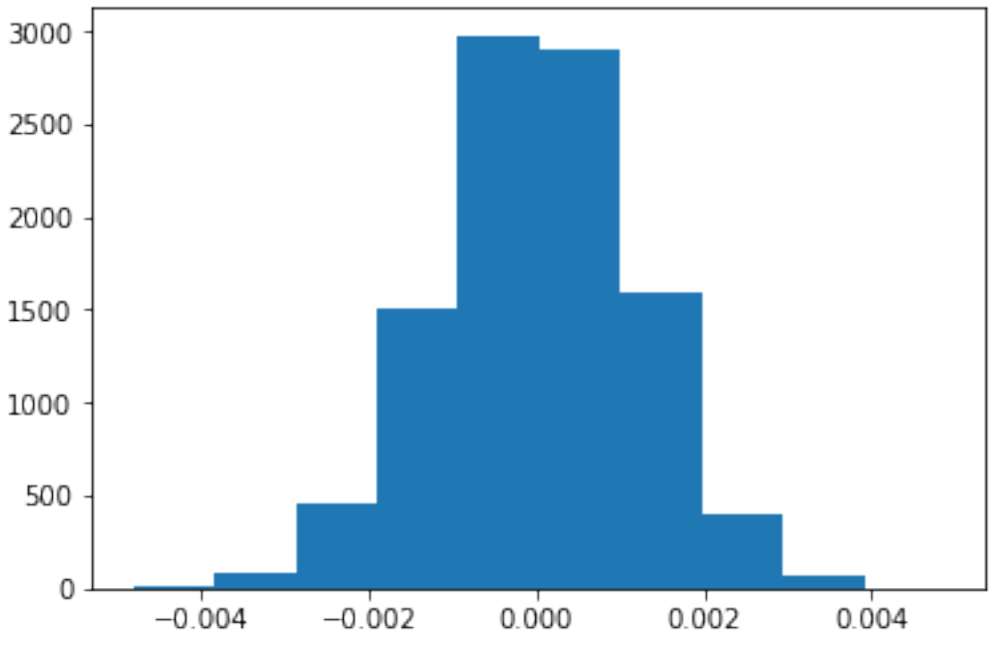
In [50]:

```
plt.hist(p_diffs);
```



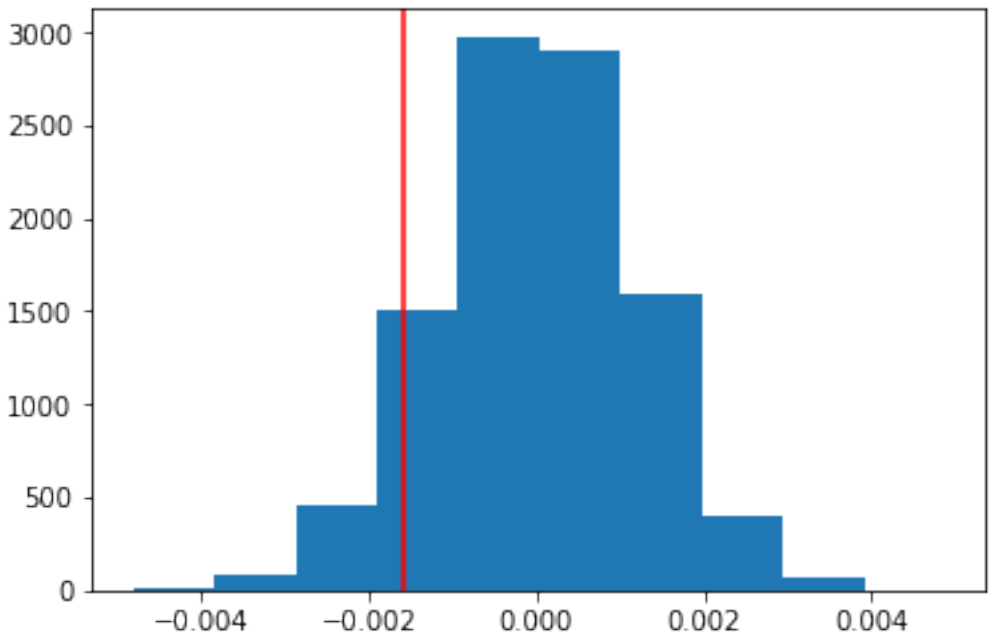
In [93]:

```
obs_diff=experiment_probability-control_probability
p_diffs=np.array(p_diffs)
samp_dist=np.random.normal(0,p_diffs.std(),p_diffs.size)
plt.hist(samp_dist);
```



In [94]:

```
plt.hist(samp_dist);
plt.axvline(x=obs_diff, color='red');
```



j. What proportion of the **p_diffs** are greater than the actual difference observed in **ab_data.csv**?

In [95]:

```
(samp_dist>obs_diff).mean()
```

Out[95]:

```
0.9081
```

In [262]:

```
(samp_dist<obs_diff).mean()+(samp_dist>obs_diff+(samp_dist-obs_diff)).mean() # a two-tailed test (optional)
```

Out[262]:

```
0.3346
```

k. In words, explain 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?

We've received a p-value of 0.9081, which means the probability of observing our statistic is 90.81% if the null hypothesis is true. The difference in conversions for control and experiment groups is not significant, therefore we fail to reject the null hypothesis.

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 [258]:

```
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
n_new = n_new
```

Out[258]:

```
(17489, 17264, 145274, 145310)
```

m. Now use `stats.proportions_ztest` to compute your test statistic and p-value. Here (http://knowledgetack.com/python/statsmodels/proportions_ztest/) is a helpful link on using the built in.

In [259]:

```
z_score, p_value = sm.stats.proportions_ztest([convert_new, convert_old], [n_new, n_old],alternative='larger')
z_score,p_value
```

Out[259]:

```
(-1.3109241984234394, 0.9050583127590245)
```

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.**?

We've got the same result, again we fail to reject the null hypothesis.
Since the p-value is 0.9050, we fail to reject the null hypothesis.

Part III - A regression approach

1. In this final part, you will see that the result you acheived in the previous A/B test can also be acheived 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?

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 a colun 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 [125]:

```
df2[['dummy','ab_page']]=pd.get_dummies(df2['group'])
df2.drop(columns=['dummy'],inplace=True)
df2['intercept']=1
df2.head()
```

Out[125]:

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

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

In [135]:

```
from scipy import stats
stats.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df) #fixing error
regression=sm.Logit(df2['converted'],df2[['intercept','ab_page']]).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 [136]:

```
regression.summary()
```

Out[136]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290584
Model:	Logit	**Df Residuals:**	290582
Method:	MLE	**Df Model:**	1
Date:	Tue, 03 Apr 2018	**Pseudo R-squ.:**	8.077e-06
Time:	18:42:22	**Log-Likelihood:**	-1.0639e+05
converged:	True	**LL-Null:**	-1.0639e+05
		LLR p-value:	0.1899

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

In [138]:

```
np.exp(-0.0150)
```

Out[138]:

```
0.9851119396030626
```

In [212]:

```
1/np.exp(-0.0150) # Change the direction. That means the control group is 1.01
5 times likely to be converted.
```

Out[212]:

```
1.015113064615719
```

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

These suggest that ab_page(p=0.190) isn't statistically significant in predicting whether or not an individual converts. Conversion is 0.98 times as likely on new page than on old page holding all else constant.Which means the same as in part 2 that the treatment group isn't better the control group. Our null hypothesis (from regression model) states that the conversion is equal zero and the alternative hypothesis isn't equal zero. In our case the probability of observing that the treatment group is differs from the control group is 0.190 if the null hypothesis is true. Therefore we fail to reject the null.

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?

In some cases, adding more factors to our regression model can improve the model prediction and accuracy. Additional factors can show if the factors have influence on the result(in our case, whether or no an individual converts). The important thing to keep in mind when adding new factors to the model, check if that factors are not correlated with each other.

g. Now along with testing if the conversion rate changes for different pages, also add an effect based on which country a user lives. You will need to read in the **countries.csv** dataset and merge together your datasets on the approporiate rows. Here (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 varaibles.** Provide the statistical output as well as a written response to answer this question.

In [198]:

```
df_countries=pd.read_csv('countries.csv')
df2=df2.merge(df_countries,on='user_id', how='inner')
df2.head()
```

Out[198]:

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

In [199]:

```
df2=df2.join(pd.get_dummies(df2['country']))
df2.head()
```

Out[199]:

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

In [223]:

```
Log_model=sm.Logit(df2['converted'],df2[['intercept','ab_page','CA','UK']]).fi
t()
Log_model.summary()
```

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

Out[223]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290584
Model:	Logit	**Df Residuals:**	290580
Method:	MLE	**Df Model:**	3
Date:	Wed, 04 Apr 2018	**Pseudo R-squ.:**	2.323e-05
Time:	12:46:48	**Log-Likelihood:**	-1.0639e+05
converged:	True	**LL-Null:**	-1.0639e+05
		LLR p-value:	0.1760

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
intercept	-1.9893	0.009	-223.763	0.000	-2.007	-1.972
ab_page	-0.0149	0.011	-1.307	0.191	-0.037	0.007
CA	-0.0408	0.027	-1.516	0.130	-0.093	0.012
UK	0.0099	0.013	0.743	0.457	-0.016	0.036

In [224]:

```
np.exp(-0.0149),np.exp(-0.0408),np.exp(0.0099)
```

Out[224]:

```
(0.9852104557227469, 0.9600211149716509, 1.0099491671175422)
```

Does it appear that country had an impact on conversion?
No. According to the p-values(p(CA)=0.130,p(UK)=0.457), we can conclude that these factors aren't statistically significant in predicting whether or not an individual converts.

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 [204]:

```
df2['inter_ca']=df2['CA']*df2['ab_page']
df2['inter_us']=df2['US']*df2['ab_page']
df2['inter_uk']=df2['UK']*df2['ab_page']
```

In [227]:

```
inter_countries=sm.Logit(df2['converted'],df2[['intercept','ab_page','CA','UK'
,'inter_ca','inter_uk']]).fit()
inter_countries.summary()
```

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

Out[227]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290584
Model:	Logit	**Df Residuals:**	290578
Method:	MLE	**Df Model:**	5
Date:	Wed, 04 Apr 2018	**Pseudo R-squ.:**	3.482e-05
Time:	12:49:04	**Log-Likelihood:**	-1.0639e+05
converged:	True	**LL-Null:**	-1.0639e+05
		LLR p-value:	0.1920

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
intercept	-1.9865	0.010	-206.344	0.000	-2.005	-1.968
ab_page	-0.0206	0.014	-1.505	0.132	-0.047	0.006
CA	-0.0175	0.038	-0.465	0.642	-0.091	0.056
UK	-0.0057	0.019	-0.306	0.760	-0.043	0.031
inter_ca	-0.0469	0.054	-0.872	0.383	-0.152	0.059
inter_uk	0.0314	0.027	1.181	0.238	-0.021	0.084

In [210]:

```
np.exp(-0.0206),np.exp(-0.0175),np.exp(-0.0057),np.exp(-0.0469),np.exp(0.0314)
```

Out[210]:

```
(0.9796107305032505,
 0.9826522356650732,
 0.9943162141784332,
 0.9541828111007262,
 1.0318981806179213)
```

According to the p-values, these suggest that interactions between page and country aren't statistically significant in predicting whether or not an individual converts.

Conclusions

Congratulations on completing the project!

Gather Submission Materials

Once you are satisfied with the status of your Notebook, you should save it in a format that will make it easy for others to read. You can use the **File -> Download as -> HTML (.html)** menu to save your notebook as an .html file. If you are working locally and get an error about "No module name", then open a terminal and try installing the missing module using `pip install <module_name>` (don't include the "<" or ">" or any words following a period in the module name).

You will submit both your original Notebook and an HTML or PDF copy of the Notebook for review. There is no need for you to include any data files with your submission. If you made reference to other websites, books, and other resources to help you in solving tasks in the project, make sure that you document them. It is recommended that you either add a "Resources" section in a Markdown cell at the end of the Notebook report, or you can include a `readme.txt` file documenting your sources.

Submit the Project

When you're ready, click on the "Submit Project" button to go to the project submission page. You can submit your files as a .zip archive or you can link to a GitHub repository containing your project files. If you go with GitHub, note that your submission will be a snapshot of the linked repository at time of submission. It is recommended that you keep each project in a separate repository to avoid any potential confusion: if a reviewer gets multiple folders representing multiple projects, there might be confusion regarding what project is to be evaluated.

It can take us up to a week to grade the project, but in most cases it is much faster. You will get an email once your submission has been reviewed. If you are having any problems submitting your project or wish to check on the status of your submission, please email us at dataanalyst-project@udacity.com. In the meantime, you should feel free to continue on with your learning journey by continuing on to the next module in the program.