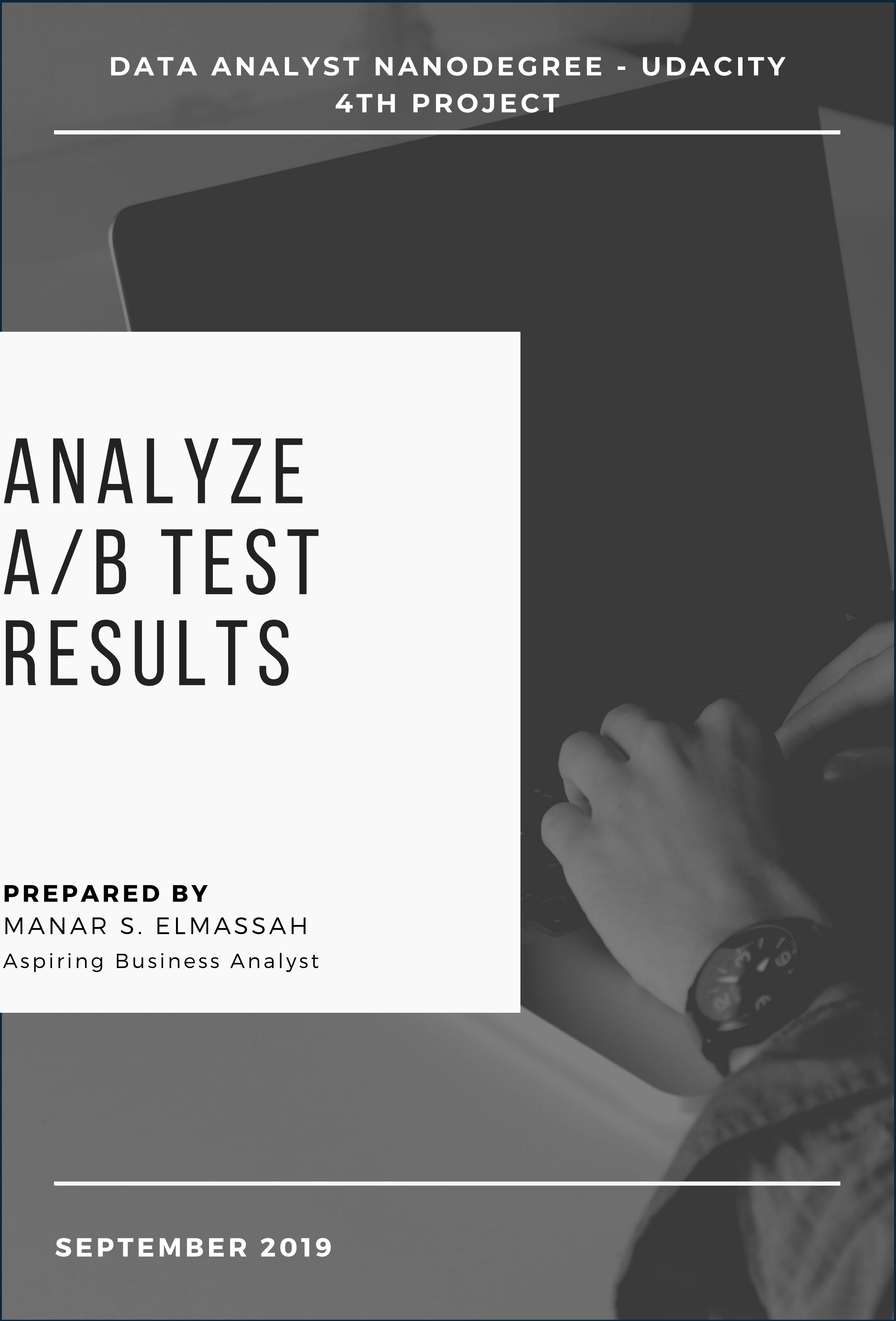


DATA ANALYST NANODEGREE - UDACITY
4TH PROJECT

# ANALYZE A/B TEST RESULTS

**PREPARED BY**
MANAR S. ELMASSAH
Aspiring Business Analyst

SEPTEMBER 2019

# 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. **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.

#### Part I - Probability

To get started, let's import our libraries.

In [18]:

```
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 [19]:

```
#Import the dataset
df=pd.read_csv("ab_data.csv")
df.head()
```

Out[19]:

| | 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 [20]:

```
#Overview of the DS
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
```

c. The number of unique users in the dataset.

In [21]:

```
df.nunique()
```

Out[21]:

```
user_id         290584
timestamp       294478
group                2
landing_page         2
converted            2
dtype: int64
```

d. The proportion of users converted.

In [22]:

```
df['converted'].mean()
```

Out[22]:

```
0.11965919355605512
```

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

In [23]:

```
x=df[(df['group'] == "treatment")&(df['landing_page'] !='new_page') ].count()
y=df[(df['group'] == "control")&(df['landing_page'] =='new_page') ].count()

x+y
```

Out[23]:

```
user_id         3893
timestamp       3893
group           3893
landing_page    3893
converted       3893
dtype: int64
```

f. Do any of the rows have missing values?

In [24]:

```
df.isnull().sum()
```

Out[24]:

```
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 [25]:

```
df_treat = df[(df['group'] == "treatment")&(df['landing_page'] !='old_page') ]
df_control = df[(df['group'] == "control")&(df['landing_page'] =='old_page') ]
frames=(df_treat,df_control)
df2=pd.concat(frames)
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 290585 entries, 2 to 294476
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 [26]:

```
# 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[26]:

```
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 [27]:

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

Out[27]:

```
290584
```

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

In [28]:

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

Out[28]:

|  | user_id | timestamp | group | landing_page | converted |
|---|---|---|---|---|---|
| **1899** | 773192 | 2017-01-09 05:37:58.781806 | treatment | new_page | 0 |

| | 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 [29]:

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

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2 entries, 1899 to 2893
Data columns (total 5 columns):
user_id         2 non-null int64
timestamp       2 non-null object
group           2 non-null object
landing_page    2 non-null object
converted       2 non-null int64
dtypes: int64(2), object(3)
memory usage: 96.0+ bytes
```

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

In [30]:

```
df2 = df2.drop_duplicates(subset='user_id', keep='first')
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 290584 entries, 2 to 294476
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 [31]:

```
df2['converted'].mean()
```

Out[31]:

```
0.11959708724499628
```

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

In [32]:

```
df2[df2['group']== 'control']['converted'].mean()
```

Out[32]:

```
0.1203863045004612
```

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

In [33]:

```
df2[df2['group']== 'treatment' ]['converted'].mean()
```

Out[33]:

```
0.11880806551510564
```

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

In [34]:

```
df2.query('landing_page == "new_page"').user_id.size / df2.user_id.size
```

Out[34]:

```
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.

**from what can be seen through the probability that the conversion rate on the old page is 0.1203 while on the new page is 0.1189 so up until now it seems the new treatment page leads to less conversion**

## 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 Hypotheses : **$p_{old}$** >= **$p_{new}$**

Alternative Hypotheses : **$p_{new}$** > **$p_{old}$**

`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.

In [35]:

```
df2.head()
```

Out[35]:

| | user_id | timestamp | group | landing_page | converted |
|---|---|---|---|---|---|
| **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 |

| | user_id | timestamp | group | landing_page | converted |
|---|---|---|---|---|---|
| **6** | 679687 | 2017-01-19 03:26:46.940749 | treatment | new_page | 1 |
| **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 |

a. What is the **conversion rate** for **$p_{new}$** under the null?

In [36]:

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

Out[36]:

```
0.11959708724499628
```

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

In [37]:

```
p_old= df2['converted'].mean()
p_old
```

Out[37]:

```
0.11959708724499628
```

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

In [38]:

```
n_new =len(df2.query("group == 'treatment'"))
n_new
```

Out[38]:

```
145310
```

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

In [39]:

```
n_old = len(df2.query("group == 'control'"))
n_old
```

Out[39]:

```
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 [40]:

```
new_page_converted= np.random.binomial(1,p_new,size=n_new)

new_page_converted
```

Out[40]:

```
array([0, 1, 0, ..., 0, 0, 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 [41]:

```
old_page_converted= np.random.binomial(1,p_old,size=n_old)

old_page_converted
```

Out[41]:

```
array([1, 0, 0, ..., 0, 0, 0])
```

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

In [42]:

```
## The mean of conversion on the old page
old_cmean=old_page_converted.mean()
## The mean of conversion on the new page
new_cmean=new_page_converted.mean()
##The difference in means
obs_diff=new_cmean-old_cmean
old_cmean,new_cmean,obs_diff
```

Out[42]:

```
(0.11838319313848314, 0.11883559287041498, 0.00045239973193184069)
```

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 [43]:

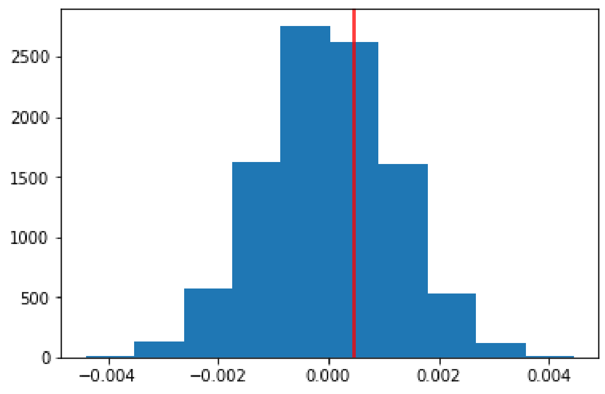
```
p_diffs = []

for _ in range(10000):
    con_sample=  np.random.binomial(1,p_old,size=n_old)
    exp_sample=  np.random.binomial(1,p_new,size=n_new)
    con_mean=con_sample.mean()
    exp_mean=exp_sample.mean()
    p_diffs.append(exp_mean-con_mean)
```

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 [44]:

```
# convert to numpy array
p_diffs = np.array(p_diffs)
# plot distribution
plt.hist(p_diffs);
plt.axvline(obs_diff, color='red');
```



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

In [45]:

```
p_value=(p_diffs>obs_diff).mean()
p_value
```

Out[45]:

```
0.35189999999999999
```

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?

**As answered in J it's found that the P_value is more than 5% which means we failed to reject the null hyposesis**

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 [46]:

```
import statsmodels.api as sm

convert_old = df2[df2['group']== 'control']['converted'].sum()
convert_new = df2[df2['group']== 'treatment']['converted'].sum()

print(convert_old,n_old ,convert_new,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 instead.
  from pandas.core import datetools
```

```
17489 145274 17264 145310
```

m. Now use `stats.proportions_ztest` to compute your test statistic and p-value. Here is a helpful link on using the built in.

In [47]:

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

Out[47]:

```
(-1.3109241984234394, 0.90505831275902449)
```

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 is lower than the critical value,So that means we fail to reject the null. and the p-values are slightly different but there is no statistically significant difference betweem the new and the old pages.**

## 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?

## 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 [48]:

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

Out[48]:

| | user_id | timestamp | group | landing_page | converted | intercept | control | treatment |
|---|---|---|---|---|---|---|---|---|
| **2** | 661590 | 2017-01-11 16:55:06.154213 | treatment | new_page | 0 | 1 | 0 | 1 |
| **3** | 853541 | 2017-01-08 18:28:03.143765 | treatment | new_page | 0 | 1 | 0 | 1 |
| **6** | 679687 | 2017-01-19 03:26:46.940749 | treatment | new_page | 1 | 1 | 0 | 1 |
| **8** | 817355 | 2017-01-04 17:58:08.979471 | treatment | new_page | 1 | 1 | 0 | 1 |
| **9** | 839785 | 2017-01-15 18:11:06.610965 | treatment | new_page | 1 | 1 | 0 | 1 |

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 [49]:

```
import statsmodels.api as sm

logit = sm.Logit(df2['converted'],df2[['intercept' ,'treatment']])
result = logit.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 [50]:

```
result.summary()
```

Out[50]:

Logit Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | converted | **No. Observations:** | 290584 |
| **Model:** | Logit | **Df Residuals:** | 290582 |
| **Method:** | MLE | **Df Model:** | 1 |
| **Date:** | Tue, 24 Sep 2019 | **Pseudo R-squ.:** | 8.077e-06 |
| **Time:** | 11:50:21 | **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 |
| **treatment** | -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**?

**p-value for ab_page is 0.19**

**The difference from part is that's a two-tailed test. we still fail to reject the null as the value is still higher than 0.05 (alpha)**

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 other factors in consideration would improve our overview of the model. but at the same time they may present more issues as outliers that might decrease the confidence rate in the 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 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 [51]:

```
#Import Data set
count_df = pd.read_csv('./countries.csv')
#merge tables
df3 = count_df.set_index('user_id').join(df2.set_index('user_id'), how='inner')
df3.head()
```

Out[51]:

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

In [52]:

```
#find unique countries
count_df['country'].unique()
```

Out[52]:

```
array(['UK', 'US', 'CA'], dtype=object)
```

In [53]:

```
# Create dummy variables
df3['intercept'] = 1
df3[['CA', 'US','UK']] = pd.get_dummies(df3['country'])[['CA', 'US','UK']]
```

In [54]:

```
# Fit Linear Model And get the Results

log_new = sm.Logit(df3['converted'], df3[['CA', 'US','UK']])
result = log_new.fit()
```

```
Optimization terminated successfully.
```

```
         Current function value: 0.366116
         Iterations 6
```

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.

**p-value is 0.198 which is still higher than alpha, result : we fail to reject the null**

**as per the results we found it's better to stay with the old page.**

In [55]:

```
result.summary()
```

Out[55]:

Logit Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | converted | **No. Observations:** | 290584 |
| **Model:** | Logit | **Df Residuals:** | 290581 |
| **Method:** | MLE | **Df Model:** | 2 |
| **Date:** | Tue, 24 Sep 2019 | **Pseudo R-squ.:** | 1.521e-05 |
| **Time:** | 11:52:49 | **Log-Likelihood:** | -1.0639e+05 |
| **converged:** | True | **LL-Null:** | -1.0639e+05 |
| | | **LLR p-value:** | 0.1984 |

| | coef | std err | z | P>\|z\| | [0.025 | 0.975] |
|---|---|---|---|---|---|---|
| **CA** | -2.0375 | 0.026 | -78.364 | 0.000 | -2.088 | -1.987 |
| **US** | -1.9967 | 0.007 | -292.314 | 0.000 | -2.010 | -1.983 |
| **UK** | -1.9868 | 0.011 | -174.174 | 0.000 | -2.009 | -1.964 |

## 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'])
```