# 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 [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 [3]:

```
df.shape
```

Out[3]:

```
(294478, 5)
```

c. The number of unique users in the dataset.

In [4]:

```
df.user_id.nunique()
```

Out[4]:

```
290584
```

d. The proportion of users converted.

In [5]:

```
df.converted.mean()
```

Out[5]:

```
0.11965919355605512
```

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

In [6]:

```
print(df.query('group != "treatment" and landing_page == "new_page"').user_id.count() +
df.query('group == "treatment" and landing_page != "new_page"').user_id.count())
```

```
3893
```

f. Do any of the rows have missing values?

In [7]:

```
df.isnull().count()
```

Out[7]:

```
user_id         294478
timestamp       294478
group           294478
landing_page    294478
converted       294478
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]:

```
df2 = df.query('group == "treatment" and landing_page == "new_page"')
df2_1 = df.query('group == "control" and landing_page == "old_page"')
df2 = df2.append(df2_1)
df2.shape
```

Out[8]:

```
(290585, 5)
```

In [9]:

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

Out[9]:

```
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 [10]:

```
df2.user_id.nunique()
```

Out[10]:

```
290584
```

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

In [11]:

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

Out[11]:

| | 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 [12]:

```
df2.iloc[2893]
```

Out[12]:

```
user_id                           723335
timestamp       2017-01-15 12:29:50.410123
group                          treatment
landing_page                    new_page
converted                              0
Name: 5917, dtype: object
```

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

In [13]:

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

Out[13]:

| | 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 |
| **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 |

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 [14]:

```
df2.converted.mean()
```

Out[14]:

```
0.11959708724499628
```

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

In [15]:

```
df2.query('group == "control"').converted.mean()
```

Out[15]:

```
0.1203863045004612
```

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

In [16]:

```
df2.query('group == "treatment"').converted.mean()
```

Out[16]:

```
0.11880806551510564
```

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

In [17]:

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

Out[17]:

```
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.**

## 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 = Pnew <= Pold H1 = Pnew > Pold

`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 [18]:

```
Pnew = df2.converted.mean()
print(Pnew)
```

```
0.119597087245
```

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

In [19]:

```
Pold = df2.converted.mean()
print(Pold)
```

```
0.119597087245
```

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

In [20]:

```
Nnew = df2.query('group == "treatment"').user_id.count()
print('The number of individuals in the treatment group is',Nnew)
```

```
The number of individuals in the treatment group is 145310
```

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

In [21]:

```
Nold = df2.query('group == "control"').user_id.count()
print('The number of individuals in the control group is',Nold)
```

```
The number of individuals in the control group is 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 [22]:

```
new_page_converted = np.random.binomial(1, Pnew, Nnew)
new_page_converted.mean()
```

Out[22]:

```
0.11819558185947285
```

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 [23]:

```
old_page_converted = np.random.binomial(1, Pold, Nold)
old_page_converted.mean()
```

Out[23]:

```
0.11918856780979391
```

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

In [24]:

```
diffs = new_page_converted.mean() - old_page_converted.mean()
diffs
```

Out[24]:

```
-0.00099298595032105974
```

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**.

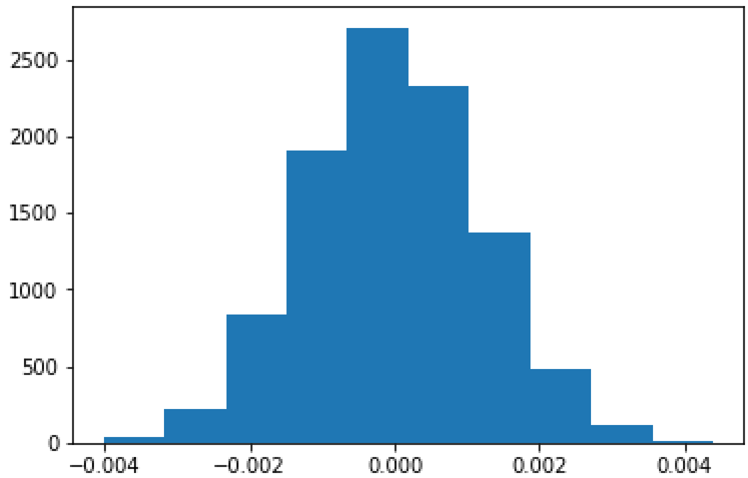
In [25]:

```
p_diffs = []
for _ in range(10000):
    oldpage_converted = np.random.binomial(1, Pold, Nold)
    newpage_converted = np.random.binomial(1, Pnew, Nnew)
    p_diffs.append(newpage_converted.mean() - oldpage_converted.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 [26]:

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




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

In [27]:

```
ccm = df2.query('group == "control"').converted.mean()
tcm = df2.query('group == "treatment"').converted.mean()
obs_diff = tcm - ccm
obs_diff
```

Out[27]:

```
-0.0015782389853555567
```

In [28]:

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

Out[28]:

```
0.9039000000000004
```

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?

The P value measures against strength of Null hypothesis. As the P value stands 0.90, which is more than threshold level of 0.05. we fail reject null hypothesis and conclude that the conversion rate for new page is less than or equal to old 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 the the number of rows associated with the old page and new pages, respectively.

In [29]:

```
import statsmodels.api as sm

convert_old = df2.query('group == "control" and converted == 1').shape[0]
convert_new = df2.query('group == "treatment" and converted == 1').shape[0]
n_old = df2.query('group == "control"').shape[0]
n_new = df2.query('group == "treatment"').shape[0]
print(convert_old, convert_new, n_old, n_new)
```

```
/opt/conda/lib/python3.6/site-packages/statsmodels/compat/pandas.py:56: Futu
reWarning: The pandas.core.datetools module is deprecated and will be remove
d in a future version. Please use the pandas.tseries module instead.
  from pandas.core import datetools

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/)](http://knowledgetack.com/python/statsmodels/proportions_ztest/) is a helpful link on using the built in.

In [30]:

```
zscore, p_value = sm.stats.proportions_ztest([convert_old,convert_new], [n_old, n_new], alt
print(zscore, p_value)
```

```
1.31092419842 0.905058312759
```

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

Yes, in the both tests p value are identical. As both tests suggest p value around 0.90 against threshold level of 0.05. we fail to reject null hypothesis.

## 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 [31]:

```
#create intercept column
df2['intercept'] = 1
```

In [32]:

```
#create dummies
ab_page = ['treatment', 'control']
df2['ab_page'] = pd.get_dummies(df2.group)['treatment']
df2.head(5)
```

Out[32]:

| | user_id | timestamp | group | landing_page | converted | intercept | ab_page |
|---|---|---|---|---|---|---|---|
| **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 |
| **6** | 679687 | 2017-01-19 03:26:46.940749 | treatment | new_page | 1 | 1 | 1 |
| **8** | 817355 | 2017-01-04 17:58:08.979471 | treatment | new_page | 1 | 1 | 1 |
| **9** | 839785 | 2017-01-15 18:11:06.610965 | treatment | new_page | 1 | 1 | 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 [33]:

```
log_mod = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])
results = log_mod.fit()
results.summary()
```

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

Out[33]:

Logit Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | converted | **No. Observations:** | 290584 |
| **Model:** | Logit | **Df Residuals:** | 290582 |
| **Method:** | MLE | **Df Model:** | 1 |
| **Date:** | Mon, 13 May 2019 | **Pseudo R-squ.:** | 8.077e-06 |
| **Time:** | 11:19:58 | **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 |

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

In [34]:

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

Out[34]:

```
1.0151130646157189
```

For each unit of decrease in ab_page, conversion is 1.015 times, which is not very significant.

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 is 0.190, compared to p value of 0.9040 and 0.9051. Since the logistic regression decide the possibility of two outcomes, it is generally considered as two tailed test unlike the previous test, which was one sided test.

H0: New page - Old page = 0 H1: New page - Old page != 0

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?

We include other varaibles that affect the responses in order to avoid the baised results and too many varaibles in the model will tend to have less precise estimates. Also, there can be Multi-collinearity between predictor variables, which will skew R values and makes interpertation difficult.

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)] 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')
countries.head()
```

Out[35]:

| | user_id | country |
|---|---|---|
| **0** | 834778 | UK |
| **1** | 928468 | US |
| **2** | 822059 | UK |
| **3** | 711597 | UK |
| **4** | 710616 | UK |

In [36]:

```
df2_countries = df2.merge(countries, on= 'user_id', how= 'inner')
df2_countries.head()
```

Out[36]:

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

In [37]:

```
df2_countries.country.unique()
```

Out[37]:

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

In [38]:

```
df2_countries[['US', 'UK']] = pd.get_dummies(df2_countries.country)[['US','UK']]
df2_countries.head()
```

Out[38]:

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

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 [39]:

```
log_mod = sm.Logit(df2_countries['converted'], df2_countries[['intercept', 'ab_page', 'US',
results = log_mod.fit()
results.summary()
```

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

Out[39]:

Logit Regression Results

| | | | |
|---|---|---|---|
| **Dep. Variable:** | converted | **No. Observations:** | 290584 |
| **Model:** | Logit | **Df Residuals:** | 290580 |
| **Method:** | MLE | **Df Model:** | 3 |
| **Date:** | Mon, 13 May 2019 | **Pseudo R-squ.:** | 2.323e-05 |
| **Time:** | 11:19:59 | **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** | -2.0300 | 0.027 | -76.249 | 0.000 | -2.082 | -1.978 |
| **ab_page** | -0.0149 | 0.011 | -1.307 | 0.191 | -0.037 | 0.007 |
| **US** | 0.0408 | 0.027 | 1.516 | 0.130 | -0.012 | 0.093 |
| **UK** | 0.0506 | 0.028 | 1.784 | 0.074 | -0.005 | 0.106 |

In [40]:

```
np.exp(0.0408), np.exp(0.0506)
```

Out[40]:

```
(1.0416437559600236, 1.0519020483004984)
```

Summary

For the US variable, the conversion is 1.041 times and for the UK variable, the conversion is 1.051 times. Both are not very significant. P value for all the variables are more than threshold of 5%. hence, we fail to reject null hypothesis. we accept that new page is not significantly better than old page.

In [41]:

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

Out[41]:

```
0
```

In [ ]:

```

```