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
- 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.

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

In [3]:
```
df['timestamp'] = pd.to_datetime(df['timestamp'])
```

In [4]:
```
print(df['group'].unique())
print(df['landing_page'].unique())
print(df['converted'].unique())
```

```
['control' 'treatment']
['old_page' 'new_page']
[0 1]
```

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

In [5]:
```
df.shape
```

Out[5]: (294478, 5)

c. The number of unique users in the dataset.

In [6]:
```
df.user_id.nunique()
```

Out[6]: 290584

d. The proportion of users converted.

In [7]:
```
df.converted.mean()
```

Out[7]: 0.11965919355605512

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

In [8]:
```
df[ (df['group'] == 'treatment') & (df['landing_page'] == 'old_page')].shape[0] + \
df[ (df['group'] == 'control') & (df['landing_page'] == 'new_page')].shape[0]
```

Out[8]: 3893

f. Do any of the rows have missing values?

In [9]:
```
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 datetime64[ns]
group           294478 non-null object
landing_page    294478 non-null object
converted       294478 non-null int64
dtypes: datetime64[ns](1), int64(2), object(2)
memory usage: 11.2+ MB
```

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

a. Now use the answer to the quiz to create a new dataset that meets the specifications from the quiz. Store your new dataframe in **df2**.

In [10]:
```
df[ (df['group'] == 'treatment') & (df['landing_page'] == 'old_page')].shape[0]
```

Out[10]: 1965

In [11]:
```
df[ (df['group'] == 'control') & (df['landing_page'] == 'new_page')].shape[0]
```

Out[11]: 1928

In [12]:
```
df[ (df['group'] == 'treatment') & (df['landing_page'] == 'new_page')].index
```

Out[12]:
```
Int64Index([     2,      3,      6,      8,      9,     10,     11,     12,
                13,     14,
            ...
            294455, 294456, 294457, 294458, 294460, 294462, 294465, 294468,
            294472, 294477],
           dtype='int64', length=145311)
```

In [13]:
```
df2 = df.drop(df[ (df['group'] == 'treatment') & (df['landing_page'] == 'old_page')].index)
```

In [14]:
```
df2 = df2.drop(df[ (df['group'] == 'control') & (df['landing_page'] == 'new_page')].index)
```

In [15]:
```
# 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[15]: 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 [16]:
```
df2.shape[0]
```

Out[16]: 290585

In [17]:
```
df2.user_id.nunique()
```

Out[17]: 290584

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

In [18]:
```
df2.user_id.value_counts().idxmax()
```

Out[18]: 773192

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

In [19]:
```
df2[df2.user_id == 773192]
```

Out[19]:

	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 [20]:
```
df2.drop(1899, inplace=True)
```

In [21]:
```
df2[df2.user_id == 773192]
```

Out[21]:

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

`4.` Use **df2** in the cells below to answer the quiz questions related to **Quiz 4** in the classroom.

a. What is the probability of an individual converting regardless of the page they receive?

In [22]:
```
df2.converted.mean()
```

Out[22]: 0.11959708724499628

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

In [23]:
```
df2.query('group == "control"').converted.mean()
```

Out[23]: 0.1203863045004612

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

In [24]:
```
df2.query('group == "treatment"').converted.mean()
```

Out[24]: 0.11880806551510564

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

In [25]:
```
df2[df2['landing_page'] == 'new_page'].shape[0] / df2.shape[0]
```

Out[25]: 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 the above results, i think that there is no sufficient evidence to conclude that the new treatment page leads to more conversions, as probability of users converted given that they are from treatment group is almost equal to probability of users converted given that they are from control.

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.

$$H_0 : p_{new} - p_{old} \leq 0$$
$$H_1 : p_{new} - p_{old} > 0$$

`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 [26]:
```
df2.converted.mean()
```

Out[26]: 0.11959708724499628

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

In [27]:
```
df2.converted.mean()
```

Out[27]: 0.11959708724499628

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

In [28]:
```
df2.query('group == "treatment"').shape[0]
```

Out[28]: 145310

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

In [29]:
```
df2.query('group == "control"').shape[0]
```

Out[29]: 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 [30]:
```
df2['converted'].unique()
```

Out[30]: array([0, 1])

In [31]:
```
new_page_converted = np.random.binomial(1, df2.converted.mean(), df2.query('group == "treatm
ent"').shape[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 [32]:
```
old_page_converted = np.random.binomial(1, df2.converted.mean(), df2.query('group == "contro
l"').shape[0])
```

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

In [33]:
```
new_page_converted.mean() - old_page_converted.mean()
```

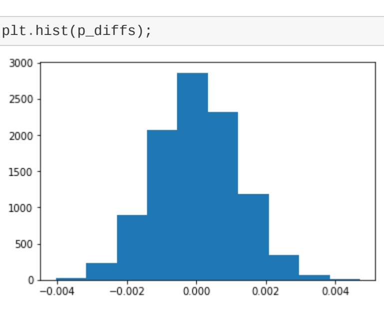
Out[33]: -0.0006560043676166194

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 [34]:
```
p_diffs = []
for _ in range(10000):
    new_page_converted = np.random.binomial(1, df2.converted.mean(), df2.query('group == "tr
eatment"').shape[0])
    old_page_converted = np.random.binomial(1, df2.converted.mean(), df2.query('group == "co
ntrol"').shape[0])
    p_diffs.append(new_page_converted.mean() - old_page_converted.mean())
p_diffs = np.array(p_diffs)
```

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.

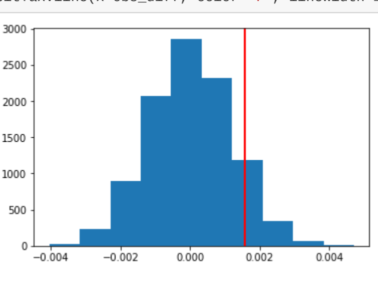
In [35]:
```
plt.hist(p_diffs);
```



In [36]:
```
obs_diff = df2.query('group == "control"').converted.mean() - df2.query('group == "treatmen
t"').converted.mean()
obs_diff
```

Out[36]: 0.0015782389853555567

In [37]:
```
plt.hist(p_diffs);
plt.axvline(x=obs_diff, color='r', linewidth=2);
```



In [38]:
```
p_diffs
```

Out[38]:
```
array([ 0.0006036 , -0.00101377,  0.00063789, ...,  0.00042426,
        0.00047963,  0.00106501])
```

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

In [39]:
```
p_valve = (p_diffs > obs_diff).mean()
p_valve
```

Out[39]: 0.09579999999999996

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?

what is mentioned in part j is called p_value, this value is greater than alpha(Type I error rate of 5%) and this means that we fail to reject the null hypothesis and we have no statistical significant evidence that new page is better than the old one so the old page is better than the new one.

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 [40]:
```
df2.query('group == "treatment"').shape[0]
```

Out[40]: 145310

In [41]:
```
df2.query('group == "treatment" and converted == 1').shape[0]
```

Out[41]: 17264

In [42]:
```
df2.query('group == "treatment" and converted == 0').shape[0]
```

Out[42]: 128046

In [44]:
```
import statsmodels.api as sm

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

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 [45]:
```
count = np.array([convert_old,convert_new])
nobs = np.array([n_old,n_new])
stat, pval = sm.stats.proportions_ztest(count, nobs)
stat, pval
```

Out[45]: (-1.3109241984234394, 0.18988337448195103)

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.**?

it confirms what we iterpret in parts j and k, pval is greater than alpha(Type I error rate of 5%) and this means that we fail to reject the null hypothesis and we have no statistical significant evidence that new page is better than the old one so the old page is better than the new one same as part j and k.

Part III - A regression approach

`1.` In this final part, you will see that the result you achieved in the A/B test in Part II above can also be achieved by performing regression.

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

In [46]:
```
df2.head()
```

Out[46]:

	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

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 [47]:
```
df2['intercept'] = 1
df2['ab_page'] = pd.get_dummies(df2['group'])['treatment']
df2.head()
```

Out[47]:

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

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

In [48]:
```
log_r = sm.Logit(df2['converted'], df2[['intercept','ab_page']])
results = log_r.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 [49]:
```
results.summary2()
```

Out[49]:

Model:	Logit	No. Iterations:	6.0000
Dependent Variable:	converted	Pseudo R-squared:	0.000
Date:	2020-05-25 07:37	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**?

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

null hypothesis refer to that there is no between new page and old page, alternative hypothesis refer to that there is a significant differance between new page and old page.

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?

time might influence individual converts, we may find relation between time intervals and individuals converts and how these interval might influence these converts, alse there might be some disadvantages as time property might depend on other properities in our model and these will lead to mis-leading results.

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 [50]:
```
df_new = pd.read_csv('countries.csv')
df_new.head()
```

Out[50]:

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

In [51]:
```
df_new[df_new['user_id'] == 679687]
```

Out[51]:

	user_id	country
176778	679687	CA

In [52]:
```
df2.head()
```

Out[52]:

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

In [53]:
```
df_merged = pd.merge(df2, df_new, how='left')
```

In [54]:
```
df_merged.head()
```

Out[54]:

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

In [55]:
```
df_merged['country'].unique()
```

Out[55]: array(['US', 'CA', 'UK'], dtype=object)

In [56]:
```
df_merged[['US','CA','UK']] = pd.get_dummies(df_merged['country'])
```

In [57]:
```
log_r = sm.Logit(df_merged['converted'], df_merged[['intercept','US','CA']])
results = log_r.fit()
```

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

In [58]:
```
results.summary2()
```

Out[58]:

Model:	Logit	No. Iterations:	6.0000
Dependent Variable:	converted	Pseudo R-squared:	0.000
Date:	2020-05-25 07:38	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	-1.9967	0.0068	-292.3145	0.0000	-2.0101	-1.9833
US	-0.0408	0.0269	-1.5178	0.1291	-0.0935	0.0119
CA	0.0099	0.0133	0.7458	0.4558	-0.0161	0.0360

In [59]:
```
np.exp(results.params)
```

Out[59]:
```
intercept    0.135779
US           0.960018
CA           1.009966
dtype: float64
```

In [60]:
```
1 / np.exp(results.params)
```

Out[60]:
```
intercept    7.364925
US           1.041647
CA           0.990133
dtype: float64
```

- p_values is greater than alpha(0.05) and that means that we fail to reject the null hypothesis and we have no statistical significant evidence that new page is different.
- also, regardless country, we predict conversion rate to be 7.36.
- aslo, as compared to a UK, we predict US to have a conversion rate by 0.960018, holding all else constant.
- on the other hand, as compared to a UK, we predict CA to have a conversion rate by 1.009966, holding all else constant.

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 [61]:
```
log_r = sm.Logit(df_merged['converted'], df_merged[['intercept','ab_page','US','CA']])
results = log_r.fit()
```

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

In [62]:
```
results.summary2()
```

Out[62]:

Model:	Logit	No. Iterations:	6.0000
Dependent Variable:	converted	Pseudo R-squared:	0.000
Date:	2020-05-25 07:38	AIC:	212781.1253
No. Observations:	290584	BIC:	212823.4439
Df Model:	3	Log-Likelihood:	-1.0639e+05
Df Residuals:	290580	LL-Null:	-1.0639e+05
Converged:	1.0000	Scale:	1.0000

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
intercept	-1.9893	0.0089	-223.7628	0.0000	-2.0067	-1.9718
ab_page	-0.0149	0.0114	-1.3069	0.1912	-0.0374	0.0075
US	-0.0408	0.0269	-1.5161	0.1295	-0.0934	0.0119
CA	0.0099	0.0133	0.7433	0.4573	-0.0162	0.0359