Mohammed Almoneef

DAND

Project 3

Analyze A/B Test Results

P3: Analyze A/B Test Results

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.

Part I - Probability
To get started, let's import our libraries.

```
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 [114]: #Read the dataset
          df = pd.read_csv('ab_data.csv')
          #Top 7 rows
          df.head(7)
```

Out[114]:

	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
5	936923	2017-01-10 15:20:49.083499	control	old_page	0
6	679687	2017-01-19 03:26:46.940749	treatment	new_page	1

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

```
In [115]: #Number of rows in dataset
          df.shape[0]

Out[115]: 294478
```

c. The number of unique users in the dataset.

```
In [116]: #Number of unique users
          df.user_id.nunique()

Out[116]: 290584
```

d. The proportion of users converted.

```
In [117]: #Proportion of users converted
          df['converted'].mean()

Out[117]: 0.11965919355605512
```

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

```
In [118]: #Identify the number of mismatches for new_page and treatment don't line up

          #Times treatment group user lands incorrectly on old_page
          treat_old = df[(df.group == 'treatment') & (df.landing_page == 'old_page')]
          treat_old.shape[0]

          #Times control group user incorrectly lands on new_page
          ctl_new = df[(df.group == 'control') & (df.landing_page == 'new_page')]
          ctl_new.shape[0]

          #Number times the new_page and treatment don't line up is
          treat_old.shape[0] + ctl_new.shape[0]

Out[118]: 3893
```

f. Do any of the rows have missing values?

P3: Analyze A/B Test Results

In [119]:
```
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
```

There are no missing value

`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**.

Delete Rows

In [8]:
```
# Delete Rows
# drop rows for mismatched treatment groups
#--df.drop(df.query("group == 'treatment' and landing_page == 'old_page'").index, inplace=True
# drop rows for mismatched control groups
#--df.drop(df.query("group == 'control' and landing_page == 'new_page'").index, inplace=True

treatment_and_new_page = (df.group == 'treatment') & (df.landing_page == 'new_page')
control_and_old_page = (df.group == 'control') & (df.landing_page == 'old_page')
clean_rows = control_and_old_page | treatment_and_new_page
df2 = df[clean_rows]
```

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 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
```

P3: Analyze A/B Test Results

```
In [10]: df2.head(7)
Out[10]:
```

	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
5	936923	2017-01-10 15:20:49.083499	control	old_page	0
6	679687	2017-01-19 03:26:46.940749	treatment	new_page	1

Double Check all of the correct rows were removed (**should be zero**)

```
In [11]: # 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[11]: 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 [12]: df2.user_id.nunique()
Out[12]: 290584
```

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

```
In [125]: sum(df2['user_id'].duplicated())
Out[125]: 1

In [126]: df2[df2['user_id'].duplicated()]['user_id']
Out[126]: 2893    773192
          Name: user_id, dtype: int64
```

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

P3: Analyze A/B Test Results

In [127]:
```
df2[df2.user_id == 773192]
```
Out[127]:

	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.**

`Remove on of the duplicate rows`

In [199]:
```
#Remove on of the duplicate rows
df2.drop(2893, inplace=True)
```

`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 [200]:
```
df2.converted.sum() / len(df2)
```
Out[200]: 0.11959708724499628

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

In [144]:
```
control_group = df2.group == 'control'
control_group_and_converted = control_group & (df2.converted == 1)
len(df2[control_group_and_converted]) / len(df2[control_group])
```
Out[144]: 0.1203863045004612

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

In [145]:
```
treatment_group = df2.group == 'treatment'
treatment_group_and_converted = treatment_group & (df2.converted == 1)
len(df2[treatment_group_and_converted]) / len(df2[treatment_group])
```
Out[145]: 0.11880806551510564

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

```
In [146]: len(df2[treatment_group]) / len(df2)

Out[146]: 0.5000619442226688
```

e. Consider your results from a. through d. above, and explain below whether you think there is sufficient evidence to say that the new treatment page leads to more conversions.

Evidence that one page leads to more conversions? Given that an individual was in the treat- ment group, the probability they converted is 0.118807. Given that an individual was in the control group, the probability they converted is 0.120386 We find that old page does better, but by a very tiny margin.Change aversion, test span durations and other potentially influencing factors are not ac-

counted for. So, we cannot state with certainty that one page leads to more conversions. This is even more important due to almost similar perforamnce of both pages

Part II - A/B Test

`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.**

Answer:

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.

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

```
In [147]: p_new = df2['converted'].mean()
          print(p_new)

          0.11959708724499628
```

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

```
In [148]: p_old = df2['converted'].mean()
          print(p_old)

          0.11959708724499628
```

c. What is n_{new}nnew?

```
In [149]: n_new = len(df2.query("group == 'treatment'"))
          print(n_new)

          145310
```

d. What is n_{old}nold?

```
In [150]: n_old = len(df2.query("group == 'control'"))
          print(n_old)

          145274
```

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

```
new_page_converted = np.random.choice([1, 0], size=n_new, p=[p_new, (1-
p_new)])
```

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

```
old_page_converted = np.random.choice([1, 0], size=n_old, p=[p_old, (1-
p_old)])
```

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

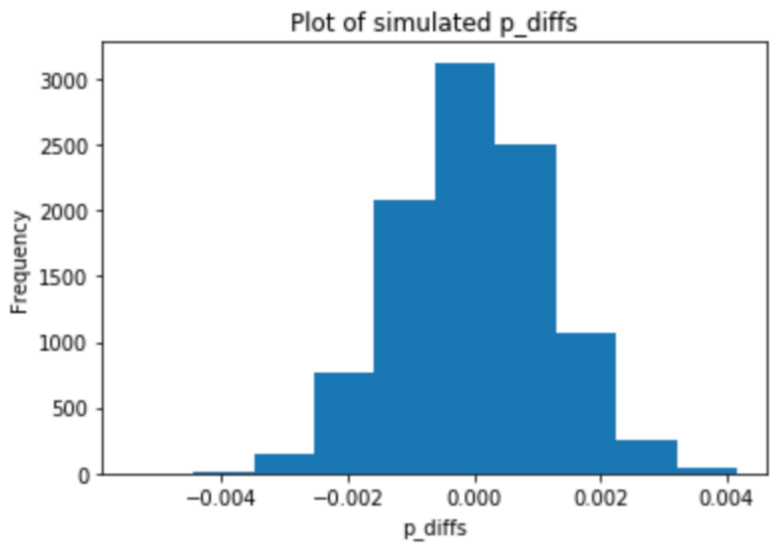
```
In [153]: new_page_converted = new_page_converted[:145274]

In [154]: p_diff = (new_page_converted/n_new) - (old_page_converted/n_old)
```

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

```
p_diffs = []
for _ in range(10000):
    new_page_converted = np.random.binomial(1, p_new, n_new)
    old_page_converted = np.random.binomial(1, p_old, n_old)
    new_page_p = new_page_converted.mean()
    old_page_p = old_page_converted.mean()
    p_diffs.append(new_page_p - old_page_p)
```

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 [99]: plt.hist(p_diffs)
         plt.xlabel('p_diffs')
         plt.ylabel('Frequency')
         plt.title('Plot of simulated p_diffs');
```



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

```
In [100]: act_diff = df[df['group'] == 'treatment']['converted'].mean() - df[df['group'] == 'control']['converted'].mean()
          act_diff
Out[100]: -0.0014795997940775518

In [101]: p_diffs = np.array(p_diffs)
          p_diffs
Out[101]: array([-2.36213905e-04,  2.42763073e-03, -1.36472462e-03, ...,
                  2.08321306e-03, -4.56390113e-04, -7.78582805e-05])

In [102]: (act_diff < p_diffs).mean()
Out[102]: 0.8903
```

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?

Answer: The p-value calculated = 0.904 (0.904 > alpha 0.5)¶

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.

```
import statsmodels.api as sm
#Number of conversions for each page
onvert_old = df2.query('group == "control" & converted == 1')['converte
d'].count()
convert_new = df2.query('group == "treatment" & converted == 1')['conve
rted'].count()
#Number of individuals who received each page
n_old = df2.query("group == 'control'")['user_id'].count()
n_new = df2.query("group == 'treatment'")['user_id'].count()
#Convert figures to integers
n_old = int(n_old)
n_new = int(n_new)
```

In [104]:
```
import statsmodels.api as sm
df2.head(7)
```

Out[104]:

	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
5	936923	2017-01-10 15:20:49.083499	control	old_page	0
6	679687	2017-01-19 03:26:46.940749	treatment	new_page	1

In [105]:
```
convert_old = sum(df2.query("group == 'control'")['converted'])
convert_new = sum(df2.query("group == 'treatment'")['converted'])
n_old = len(df2.query("group == 'control'"))
n_new = len(df2.query("group == 'treatment'"))
```

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 [106]:
```
z_score, p_value = sm.stats.proportions_ztest([convert_old, convert_new], [n_old, n_new])
print(z_score, p_value)
```

```
1.3109241984234394 0.18988337448195103
```

In [107]:
```
from scipy.stats import norm
norm.cdf(z_score)
```

Out[107]: 0.9050583127590245

In [108]:
```
norm.ppf(1-(0.05/2)) # critical value at 95% confidence
```

Out[108]: 1.959963984540054

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.**?

Answer: We will consider that we want more than 95 % confidence in this cocnclusion. So, the 1.31 Z score is less than the critical value. Reject H0 that the new page has a conversion rate no better than the old page.(An alpha level of 0.05 indicates that we have a 5% chance of committing a Type I error if the null is true.) So, we will fail to reject the null & conclude that there is no evidence to say that there is a difference between the two values.

P3: Analyze A/B Test Results

Part III - A regression approach

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?

Answer: 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 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**.

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

Out[109]:

	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 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 [110]:
```
log_mod = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])
result = log_mod.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 [114]: result.summary()
```

Out[114]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290584
Model:	Logit	**Df Residuals:**	290582
Method:	MLE	**Df Model:**	1
Date:	Sun, 05 May 2019	**Pseudo R-squ.:**	8.077e-06
Time:	17:38:55	**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

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

Answer: Our hypothesis here is: H0 : Pnew - Pold = 0, H1 : Pnew - Pold != 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?

Answer: I think that will be interesting to have a look to correlation between participants behaviors for the colors of the web page. We can check there gender and main reasons why they want to use and visit our website. For example, a child wants to play video game in our website to make friends, have fun, try somting new, etc.. .The main advantage here is that it will help us to get some ideas or make decisions to attract and get more viewers to click our website. The main problem to adding more additional terms in my regression model, it would look messy.

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]() are the docs for joining tables.

In [115]:
```
countries_df = pd.read_csv('./countries.csv')
df_new = countries_df.set_index('user_id').join(df2.set_index('user_id'), how='inner')
```

In [116]:
```
### Create the necessary dummy variables
countries = pd.read_csv('countries.csv')
countries.head()
```

Out[116]:

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

In [117]:
```
new = countries.set_index('user_id').join(df2.set_index('user_id'), how = 'inner')
new.head()
```

Out[117]:

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

In [118]:
```
new[['US', 'UK']] = pd.get_dummies(new['country'])[['US', "UK"]]
new.head()
```

Out[118]:

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

In [119]:
```
new['US_ab_page'] = new['US']*new['ab_page']
new.head()
```

Out[119]:

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

In [120]:
```
new['UK_ab_page'] = new['UK']*new['ab_page']
new.head()
```

Out[120]:

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

In [121]:
```
logit3 = sm.Logit(new['converted'], new[['intercept', 'ab_page', 'US', 'UK', 'US_ab_page', 'UK_ab_page']])
logit3
```

Out[121]: <statsmodels.discrete.discrete_model.Logit at 0x116cf8eb8>

In [122]:
```
result3 = logit3.fit()
```

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

In [123]:
```
### Fit Your Linear Model And Obtain the Results
result3.summary()
```

Out[123]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290584
Model:	Logit	**Df Residuals:**	290578
Method:	MLE	**Df Model:**	5
Date:	Sun, 05 May 2019	**Pseudo R-squ.:**	3.482e-05
Time:	17:38:57	**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	-2.0040	0.036	-55.008	0.000	-2.075	-1.933
ab_page	-0.0674	0.052	-1.297	0.195	-0.169	0.034
US	0.0175	0.038	0.465	0.642	-0.056	0.091
UK	0.0118	0.040	0.296	0.767	-0.066	0.090
US_ab_page	0.0469	0.054	0.872	0.383	-0.059	0.152
UK_ab_page	0.0783	0.057	1.378	0.168	-0.033	0.190

```
In [125]: np.exp(result.params)

Out[125]: intercept    0.136863
          ab_page      0.985123
          dtype: float64

In [126]: 1/_

Out[126]: 0.00010001000100010001

In [127]: df.groupby('group').mean()['converted']

Out[127]: group
          control      0.120399
          treatment    0.118920
          Name: converted, dtype: float64
```
