Analyze A/B Test Results

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.

IMPORT AND VIEW DATA

[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)
```

+ Code   + Markdown

`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:

[2]:
```
df=pd.read_csv('../input/ab-test/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

CONTROL DATA – (MISSING ROWS, SHAPE ETC.)

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

[3]:
```
df.shape[0]
```

Out[3]:
```
294478
```

c. The number of unique users in the dataset.

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

d. The proportion of users converted.

[4]:
```
df['converted'].mean()
```

Out[4]:
```
0.11965919355605512
```

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

```
df.groupby(['group','landing_page']).count()
#control group should have old page but there are 1928 people in control group who received new page
#treatment group should have new page but there are 1965 people in control group who received old page
```

Out[5]:

group	landing_page	user_id	timestamp	converted
control	new_page	1928	1928	1928
	old_page	145274	145274	145274
treatment	new_page	145311	145311	145311
	old_page	1965	1965	1965

[6]:
```
df.query('group == "control" and landing_page == "new_page"').nunique()+df.query('group == "treatment" and landing_page == "old_page"').nunique()
#total 3893 people received incorrect page
```

Out[6]:
```
user_id         3893
timestamp       3893
group              2
landing_page       2
converted          4
dtype: int64
```

f. Do any of the rows have missing values?

[7]:
```
df.isnull().sum()
#There is not missing value for columns
```

Out[7]:
```
user_id         0
timestamp       0
group           0
landing_page    0
converted       0
dtype: int64
```

CREATE DF2 WITH CORRECT VALUES

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

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

[8]:
```
# I kept where group is treatment and page is new + group is control and page is old under df2 dataset
df2 = df[((df['group'] == 'treatment') == (df['landing_page'] == 'new_page'))]
df2.head()
```

Out[8]:

	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

[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[0]
```

Out[9]:
```
0
```

FIND AND DELETE DUPLICATED ROWS

`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**?

[10]:
```
df2['user_id'].nunique()
```

Out[10]:
```
290584
```

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

[11]:
```
#there is 1 user_id repeated in df2
df2['user_id'].duplicated().sum()
```

Out[11]:
```
1
```

[12]:
```
print(df2[df2['user_id'].duplicated()])
#user_id 773192 is repeated at line 2893
```

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

[13]:
```
print(df2[df2['user_id'].duplicated()])
```

```
      user_id                   timestamp      group landing_page  converted
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**.

[14]:
```
#drop dublicates and save dataset to df2
df2=df2.drop_duplicates(subset=['user_id'])
```

[15]:
```
#control if 2893.line is deleted
df2.query('user_id=="773192"')
```

Out[15]:

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

CALCULATE CONVERSION RATES OF OVERALL, CONTROL GROUP AND TREATMENT GROUP

`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?

+ Code + Markdown

[16]:
```
#converted percantage
df2['converted'].mean()
```

Out[16]:
```
0.11959708724499628
```

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

[17]:
```
df2.query('group == "control"')['converted'].mean()
```

Out[17]:
```
0.1203863045004612
```

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

[18]:
```
df2.query('group == "treatment"')['converted'].mean()
```

Out[18]:
```
0.11880806551510564
```

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

+ Code + Markdown

```
df2.groupby(['landing_page']).count()
#there are 145310 people who received new page
```

Out[19]:

	user_id	timestamp	group	converted
landing_page				
new_page	145310	145310	145310	145310
old_page	145274	145274	145274	145274

[20]:
```
#percantage of people who received new page
#formula: new page / new page + old page
df2.query('landing_page=="new_page"').count()/len(df2)
```

Out[20]:
```
user_id         0.500062
timestamp       0.500062
group           0.500062
landing_page    0.500062
converted       0.500062
dtype: float64
```

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.

Your answer goes here.

My comment:

overall conversation: 0.1196 control group conversation: 0.1203 (%7 more than overall) treatment group conversation: 0.1189 (8% less than overall)

8% is small percantage. I can not say that new page causes more conversations. Old page seems better than new page at this line. We should do more tests to understand better if new page is really unsuccessful.

[21]:
```
df2[df2['group']=='treatment'].timestamp.max(), df2[df2['group']=='treatment'].timestamp.min()
```

Out[21]:
```
('2017-01-24 13:41:44.097174', '2017-01-02 13:42:05.378582')
```

My comment:

Test is run for 22 days. This might be short time for users to understand if website is better or not.

DEFINE NULL AND ALTERNATIVE HYPOTHESES

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.

+ Code + Markdown

Put your answer here.

My comment:

My null hypotheses: H0= new page is unsuccessful or as successfull as old page My alternative hypotheses: H1 = new page is more successfull than old page

H0: p_new-p_old<=0

H1: p_new-p_old>0

CALCULATE SUCCESS RATES

`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.

+ Code + Markdown

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

[22]:
```
p_new = df2.converted.mean()
p_new
```

Out[22]:
```
0.11959708724499628
```

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

[23]:
```
p_old = df2.converted.mean()
p_old
```

Out[23]:
```
0.11959708724499628
```

c. What is n_{new}?

c. What is n_{new}?

[24]:
```
n_new = df2.query("landing_page == 'new_page'")['converted'].count()
n_new
```

Out[24]: 145310

d. What is n_{old}?

[25]:
```
n_old = df2.query('landing_page == "old_page"')['converted'].count()
n_old
```

Out[25]: 145274

SIMULATE SUCCESS RATES WITH BINOMIAL FUNCTION

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

[26]:
```
#binomial function draws samples of normal distribution
# 1 trial, probability is p_new(0.1195) and we will do this n_new (145310) times

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

```
new_page_converted
```

Out[27]: array([0, 0, 0, ..., 0, 0, 0])

[28]:
```
new_page_converted.mean()
```

Out[28]: 0.11908333906819903

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

[29]:
```
old_page_converted = np.random.binomial(1, p_old, n_old)
```

[30]:
```
old_page_converted.mean()
```

Out[30]: 0.11889945895342593

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

[54]:
```
obs_simulation_diff = new_page_converted.mean() - old_page_converted.mean()
obs_simulation_diff.mean()
```

Out[54]: 0.00018388011477309119

+ Code + Markdown

My comment:

observed differences convert rate of pages is not significant. But we should evaluate it 10000 times with bootstraping and observe difference again.

CALCULATE SUCCESS RATES OF TREATMENT AND CONTROL GROUP

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

+ Code + Markdown

[32]:

```
df2.head()
```

Out[32]:

	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

[80]:

```
#diffs of control team converted rate and treatment team converted rate
obs_diff = df2.query('group == "treatment"')['converted'].mean() - df2.query('group == "control"')['converted'].mean()
obs_diff
```

Out[80]

```
-0.0015782389853555567
```

My comment:

Observed differences is -0.00157. Really small. Now I will evaluate differences with bootstraping method

SIMULATE SUCCESS RATES OF GROUPS WITH BOOTSTRAPING

[40]:

```
# create sampling distribution of difference in average converted rate
# with boostrapping
diffs = []
size = df2.shape[0]

for _ in range(1000):
    b_samp = df2.sample(size, replace=True)
    control_mean = b_samp.query('group == "control"').converted.mean()
    treatment_mean = b_samp.query('group == "treatment"').converted.mean()
    diffs.append(treatment_mean - control_mean)
```

+ Code + Markdown

My note:

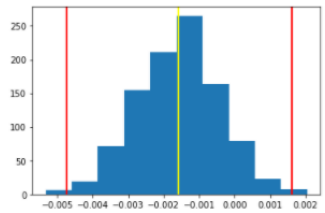
I could not arrange my range to 10000. My cpu can not compute it.

My comment:

I appended every differences to diffs array for each samples.

DRAWING HISTOGRAM

```
diffs = np.array(diffs)
# 99% confidence interval
low= np.percentile(diffs, .5)
upper=np.percentile(diffs, 99.5)
plt.hist(diffs);
plt.axvline(x=low, color='red', linewidth=2);
plt.axvline(x=upper, color='red', linewidth=2);
plt.axvline(obs_diff, color='yellow', linewidth=2);
```



+ Code + Markdown

My comment:

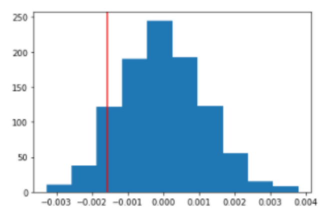
I can see bel shaped graphic, so there is normal distribution. confidence interval is %99 and almost most of samples' mean are between lower and upper line

```
null_vals =np.random.normal(0,diffs.std(), diffs.size)
plt.hist(null_vals)

# plot line for observed statistic
plt.axvline(obs_diff, c='red')
```

Out[60]

```
<matplotlib.lines.Line2D at 0x7f93b8f36a50>
```



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

[79]:

```
#p value calculation
(null_vals > obs_diff).mean()
```

Out[79]

```
0.919
```

COMMENT

My comment:

Type I error is 0.5

P value is greater than 0.5.

** We can not reject null hypotesis **