Analyze A/B Test Results

Introduction

A/B tests are very commonly performed by data analysts and data scientists. For this project, I will be working to understand the results of an A/B test run by an e-commerce website. My goal is to work through this notebook to help the company understand if they should:

- Implement the new webpage,
- Keep the old webpage, or
- Perhaps run the experiment longer to make their decision.

Part I - Probability

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)
```

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[0]
```

Out[3]:

```
294478
```

c. The number of unique users in the dataset.

In [4]:

```
df.nunique()
```

Out[4]:

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

d. The proportion of users converted.

In [5]:

```
(df['converted'] == 1).mean()
```

Out[5]:

```
0.11965919355605512
```

e. The number of times when the "group" is `treatment` but "landing_page" is not a `new_page`.

In [6]:

```
df.query('group == "treatment" and landing_page == "old_page"').shape[0]
```

Out[6]:

```
1965
```

f. Do any of the rows have missing values?

In [7]:

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

Out[7]:

```
user_id         0
timestamp       0
group           0
landing_page    0
converted       0
dtype: int64
```

In [8]:

```
# Remove the inaccurate rows, and store the result in a new dataframe df2
df2= df.query('group == "treatment" and landing_page == "new_page" or group == "control" and landing_page == "old_page"')
```

In [9]:

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

Out[9]:

```
0
```

a. How many unique **user_id**s are in **df2**?

In [10]:

```
df2.nunique()[0]
```

Out[10]:

```
290584
```

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

In [11]:

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

Out[11]:

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

c. Display the rows for the duplicate **user_id**?

In [12]:

```
df2[df2['user_id'] == 773192]
```

Out[12]:

	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**, from the **df2** dataframe.

In [13]:

```
# Remove one of the rows with a duplicate user_id..
df2.drop(2893,axis=0,inplace=True)
# Check again if the row with a duplicate user_id is deleted or not
df2[df2['user_id'] == 773192]
```

```
C:\Users\Hello\anaconda3\lib\site-packages\pandas\core\frame.py:4163: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return super().drop(
```

Out[13]:

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

In [14]:

```
(df2['converted'] == 1).mean()
```

Out[14]:

```
0.11959708724499628
```

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

In [15]:

```
p1=df2.query('group == "control" and converted == 1').shape[0]/df2[df2['group'] == 'control'].shape[0]
p1
```

Out[15]:

```
0.1203863045004612
```

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

In [16]:

```
p2=df2.query('group == "treatment" and converted == 1').shape[0]/df2[df2['group'] == 'treatment'].shape[0]
p2
```

Out[16]:

```
0.11880806551510564
```

In [17]:

```
# Calculate the actual difference (obs_diff) between the conversion rates for the two groups.
obs_diff = p1-p2
```

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

In [18]:

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

Out[18]:

```
0.5000619442226688
```

e. Considering the results from parts (a) through (d) above, explaining below whether the new `treatment` group users lead to more conversions.

No,they do not lead to more conversions

Part II - A/B Test

Since a timestamp is associated with each event, you could run a hypothesis test continuously as long as you observe the events.

However, then the hard questions would be:

- 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?

$H0$: $pold$ - $pnew$ >= 0 $H1$: $pnew$ - $pold$ > 0

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

In [19]:

```
(df2['converted'] == 1).mean()
```

Out[19]:

```
0.11959708724499628
```

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

In [20]:

```
p=(df2['converted'] == 1).mean()
p
```

Out[20]:

```
0.11959708724499628
```

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

In [21]:

```
df2['group'].value_counts()[0]
```

Out[21]:

```
145310
```

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

In [22]:

```
df2['group'].value_counts()[1]
```

Out[22]:

```
145274
```

e. Simulate Sample for the `treatment` Group
Simulate n_{new} transactions with a conversion rate of p_{new} under the null hypothesis.

Hint: Use `numpy.random.choice()` method to randomly generate n_{new} number of values.
Store these n_{new} 1's and 0's in the `new_page_converted` numpy array.

In [23]:

```
new_page_converted = []
new_page_converted = np.random.choice([0,1],size=df2['group'].value_counts()[0],p=[1-p,p])
new_page_converted = np.array(new_page_converted)
p_new = new_page_converted.mean()
```

f. Simulate Sample for the `control` Group
Simulate n_{old} transactions with a conversion rate of p_{old} under the null hypothesis.
Store these n_{old} 1's and 0's in the `old_page_converted` numpy array.

In [24]:

```
old_page_converted = []
old_page_converted = np.random.choice([0,1],size=df2['group'].value_counts()[1],p=[1-p,p])
old_page_converted = np.array(old_page_converted)
p_old = old_page_converted.mean()
```

g. Find the difference in the "converted" probability $(p'_{new}$ - $p'_{old})$ for your simulated samples from the parts (e) and (f) above.

In [25]:

```
p_old - p_new
```

Out[25]:

```
0.00037365797404279033
```

h. Sampling distribution
Re-create `new_page_converted` and `old_page_converted` and find the $(p'_{new}$ - $p'_{old})$ value 10,000 times using the same simulation process you used in parts (a) through (g) above.

Store all $(p'_{new}$ - $p'_{old})$ values in a NumPy array called `p_diffs`.

In [26]:

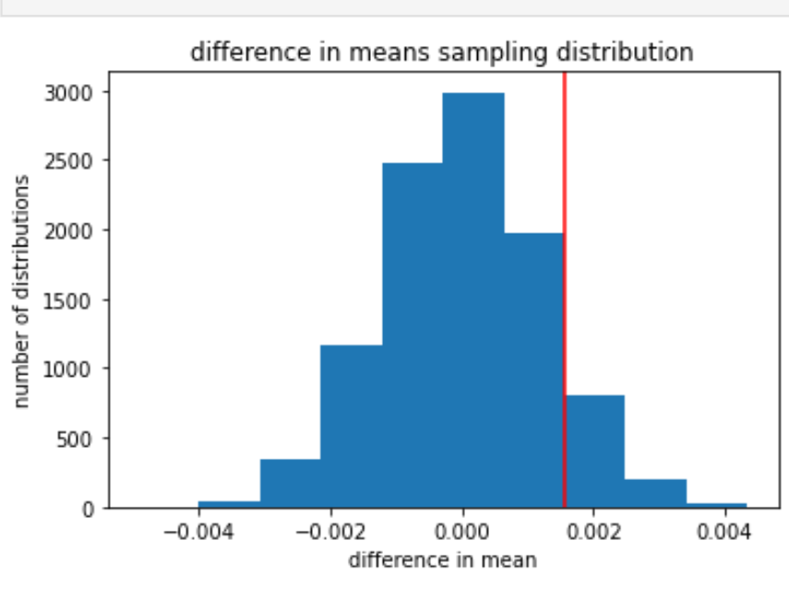
```
# Sampling distribution
p_diffs = []
for _ in range(10000):
    new_page_converted = np.random.choice([0,1],size=df2['group'].value_counts()[0],p=[1-p,p])
    p_new = np.array(new_page_converted).mean()
    old_page_converted = np.random.choice([0,1],size=df2['group'].value_counts()[1],p=[1-p,p])
    p_old = np.array(old_page_converted).mean()
    p_diffs.append(p_old - p_new)
```

In [27]:

```
p_diffs = np.array(p_diffs)
```

In [28]:

```
plt.hist(p_diffs)
plt.xlabel('difference in mean')
plt.ylabel('number of distributions')
plt.title('difference in means sampling distribution')
plt.axvline(obs_diff,color='r');
```



j. What proportion of the **p_diffs** are greater than the actual difference observed in the `df2` data?

In [29]:

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

Out[29]:

```
0.0986
```

k. Please explain in words what you have just computed in part **j** above.

- What is this value called in scientific studies?
- What does this value signify in terms of whether or not there is a difference between the new and old pages? *Hint*: Compare the value above with the "Type I error rate (0.05)".

the value is the p-value and it means that the results obtained is most likely to be generated from the null hypothesis since its larger than the alpha value stated above (0.05) so we fail to reject the null hypothesis

l. Using Built-in Methods for Hypothesis Testing
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 statements below to calculate the:

- `convert_old` : number of conversions with the old_page
- `convert_new` : number of conversions with the new_page
- `n_old` : number of individuals who were shown the old_page
- `n_new` : number of individuals who were shown the new_page

In [30]:

```
import statsmodels.api as sm

# number of conversions with the old_page
convert_old = df2.query('group == "control" and converted == 1').shape[0]

# number of conversions with the new_page
convert_new = df2.query('group == "treatment" and converted == 1').shape[0]

# number of individuals who were shown the old_page
n_old = df2['group'].value_counts()[1]

# number of individuals who received new_page
n_new = df2['group'].value_counts()[0]
```

m. Now I will use `sm.stats.proportions_ztest()` to compute my test statistic and p-value. Here is a helpful link on using the built in.

The syntax is:

```
proportions_ztest(count_array, nobs_array, alternative='larger')
```

where,

- `count_array` = represents the number of "converted" for each group
- `nobs_array` = represents the total number of observations (rows) in each group
- `alternative` = choose one of the values from `[‘two-sided’, ‘smaller’, ‘larger’]` depending upon two-tailed, left-tailed, or right-tailed respectively.

> **Hint**:
> It's a two-tailed if you defined H_1 as $(p_{new} = p_{old})$.
> It's a left-tailed if you defined H_1 as $(p_{new} < p_{old})$.
> It's a right-tailed if you defined H_1 as $(p_{new} > p_{old})$.

The built-in function above will return the z_score, p_value.

About the two-sample z-test

Recall that you have plotted a distribution `p_diffs` representing the difference in the "converted" probability $(p'_{new} - p'_{old})$ for your two simulated samples 10,000 times.

Another way for comparing the mean of two independent and normal distribution is a **two-sample z-test**. You can perform the Z-test to calculate the Z_score, as shown in the equation below:

$$Z_{score} = \frac{(p'_{new} - p'_{old}) - (p_{new} - p_{old})}{\sqrt{\frac{\sigma^2_{new}}{n_{new}} + \frac{\sigma^2_{old}}{n_{old}}}}$$

where,

- p' is the "converted" success rate in the sample
- p_{new} and p_{old} are the "converted" success rate for the two groups in the population.
- σ_{new} and σ_{new} are the standard deviation for the two groups in the population.
- n_{new} and n_{old} represent the size of the two groups or samples (it's same in our case)

> Z-test is performed when the sample size is large, and the population variance is known. The z-score represents the distance between the two "converted" success rates in terms of the standard error.

Next step is to make a decision to reject or fail to reject the null hypothesis based on comparing these two values:

- Z_{score}
- Z_{α} or $Z_{0.05}$, also known as critical value at 95% confidence interval. $Z_{0.05}$ is 1.645 for one-tailed tests, and 1.960 for two-tailed test. You can determine the Z_{α} from the z-table manually.

Decide if your hypothesis is either a two-tailed, left-tailed, or right-tailed test. Accordingly, reject OR fail to reject the null based on the comparison between Z_{score} and Z_{α}.

> Hint:
> For a right-tailed test, reject null if Z_{score} > Z_{α}.
> For a left-tailed test, reject null if Z_{score} < Z_{α}.

In other words, we determine whether or not the Z_{score} lies in the "rejection region" in the distribution. A "rejection region" is an interval where the null hypothesis is rejected iff the Z_{score} lies in that region.

Reference:

- Example 9.1.2 on this page/09%3A_Two-Sample_Problems/9.01%3A_Comparison_of_Two_Population_Means-_Large_Independent_Samples), courtesy www.stats.libretexts.org

In [31]:

```
import statsmodels.api as sm
# ToDo: Complete the sm.stats.proportions_ztest() method arguments
z_score, p_value = sm.stats.proportions_ztest([convert_old,convert_new],[n_old,n_new],alternative= 'larger')
print(z_score, p_value)
```

```
1.3109241984234394 0.09494168724097551
```

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.**?

> **Tip**: Notice whether the p-value is similar to the one computed earlier. Accordingly, can you reject/fail to reject the null hypothesis? It is important to correctly interpret the test statistic and p-value.

> **Put your answer here.** yes, they agree with the findings obtained since the test is right tailed so we fail to reject the null because the z score is less than the z alpha

Part III - A regression approach

Logistic regression

b. The goal is to use **statsmodels** library to fit the regression model you specified in part **a.** above to see if there is a significant difference in conversion based on the page-type a customer receives. However, you first need to create the following two columns in the `df2` dataframe:

1. `intercept` - It should be `1` in the entire column.
2. `ab_page` - It's a dummy variable column, having a value `1` when an individual receives the **treatment**, otherwise `0`.

In [32]:

```
df2['intercept'] = 1
df2=df2.join(pd.get_dummies(df2['group']))
df2.head()
```

```
<ipython-input-32-40a9e9c1a6d4>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df2['intercept'] = 1
```

Out[32]:

	user_id	timestamp	group	landing_page	converted	intercept	control	treatment
0	851104	2017-01-21 22:11:48.556739	control	old_page	0	1	1	0
1	804228	2017-01-12 08:01:45.159739	control	old_page	0	1	1	0
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
4	864975	2017-01-21 01:52:26.210827	control	old_page	1	1	1	0

In [33]:

```
df2.rename(columns={'treatment' : 'ab_page'},inplace=True)
df2.head()
```

Out[33]:

	user_id	timestamp	group	landing_page	converted	intercept	control	ab_page
0	851104	2017-01-21 22:11:48.556739	control	old_page	0	1	1	0
1	804228	2017-01-12 08:01:45.159739	control	old_page	0	1	1	0
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
4	864975	2017-01-21 01:52:26.210827	control	old_page	1	1	1	0

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

In [34]:

```
md = sm.Logit(df2['converted'],df2[['intercept','ab_page']])
result = md.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 [35]:

```
result.summary2()
```

Out[35]:

Model:	Logit	Pseudo R-squared:	0.000
Dependent Variable:	converted	AIC:	212780.3502
Date:	2022-05-14 20:54	BIC:	212801.5095
No. Observations:	290584	Log-Likelihood:	-1.0639e+05
Df Model:	1	LL-Null:	-1.0639e+05
Df Residuals:	290582	LLR p-value:	0.18988
Converged:	1.0000	Scale:	1.0000
No. Iterations:	6.0000		

	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**?

Hints:

- 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**?
- You may comment on if these hypothesis (Part II vs. Part III) are one-sided or two-sided.
- You may also compare the current p-value with the Type I error rate (0.05).

> **Put your answer here.** the p-value of the ab_page is 0.1899 and it differ than the value obtained before Because it is compared to the control group as baseline

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?

> **Put your answer here.** considering other factors is a good idea as time analysis or the location of the individual because only changing the page design is not enough factor to determine whether an individual converts or not but there is diadvantage that the accuracy of the model may decrease depending on the factor we are adding

g. Adding countries
Now along with testing if the conversion rate changes for different pages, also add an effect based on which country a user lives in.

1. You will need to read in the **countries.csv** dataset and merge together your `df2` datasets on the appropriate rows. You call the resulting dataframe `df_merged`. Here are the docs for joining tables.
2. Does it appear that country had an impact on conversion? To answer this question, consider the three unique values, `['UK', 'US', 'CA']`, in the `country` column. Create dummy variables for these country columns.

Provide the statistical output as well as a written response to answer this question.

In [36]:

```
# Read the countries.csv
country = pd.read_csv('countries.csv')
country.user_id.nunique()
```

Out[36]:

```
290584
```

In [37]:

```
# Join with the df2 dataframe
df_merged = df2.merge(country,on='user_id')
```

In [38]:

```
# Create the necessary dummy variables
df_merged= df_merged.join(pd.get_dummies(df_merged['country']))
df_merged.head()
```

Out[38]:

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

In [39]:

```
md = sm.Logit(df_merged['converted'],df_merged[['intercept','UK','US']])
result = md.fit()
result.summary2()
```

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

Out[39]:

Model:	Logit	Pseudo R-squared:	0.000
Dependent Variable:	converted	AIC:	212780.8333
Date:	2022-05-14 20:54	BIC:	212812.5723
No. Observations:	290584	Log-Likelihood:	-1.0639e+05
Df Model:	2	LL-Null:	-1.0639e+05
Df Residuals:	290581	LLR p-value:	0.19835
Converged:	1.0000	Scale:	1.0000
No. Iterations:	6.0000		

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
intercept	-2.0375	0.0260	-78.3639	0.0000	-2.0885	-1.9866
UK	0.0507	0.0284	1.7863	0.0740	-0.0049	0.1064
US	0.0408	0.0269	1.5178	0.1291	-0.0119	0.0935

h. Fit your model and obtain the results
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 are there significant effects on conversion. **Create the necessary additional columns, and fit the new model.**

Provide the summary results (statistical output), and your conclusions (written response) based on the result

> **Hints**:
>
> - Look at all of p-values in the summary, and compare against the Type I error rate (0.05).
> - Can you reject/fail to reject the null hypotheses (regression model)?
> - Comment on the effect of page and country to predict the conversion.

In [40]:

```
# Fit your model, and summarize the results
md = sm.Logit(df_merged['converted'],df_merged[['intercept','ab_page','UK','US']])
result = md.fit()
result.summary2()
```

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

Out[40]:

Model:	Logit	Pseudo R-squared:	0.000
Dependent Variable:	converted	AIC:	212781.1253
Date:	2022-05-14 20:54	BIC:	212823.4439
No. Observations:	290584	Log-Likelihood:	-1.0639e+05
Df Model:	3	LL-Null:	-1.0639e+05
Df Residuals:	290580	LLR p-value:	0.17599
Converged:	1.0000	Scale:	1.0000
No. Iterations:	6.0000		

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
intercept	-2.0300	0.0266	-76.2488	0.0000	-2.0822	-1.9778
ab_page	-0.0149	0.0114	-1.3069	0.1912	-0.0374	0.0075
UK	0.0506	0.0284	1.7835	0.0745	-0.0050	0.1063
US	0.0408	0.0269	1.5161	0.1295	-0.0119	0.0934

> **Put your conclusion answer here.** the p-values are too high to consider it statiscally signficant compared to the type 1 error rate (0.05) so we fail to reject the null hypothesis so as it seems the effect of the factors provided helped us to know which page is doing better but there are other factors that might affect as the product or service the client is buying

In [42]:

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

Out[42]:

```
1
```

In [ ]: