Analyze A/B Test Results

Table of Contents

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 s
et 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

Number of rows in the dataset.

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

Out[3]: 294478

Number of unique users in the dataset.

In [4]:
```
df['user_id'].nunique()
```

Out[4]: 290584

The proportion of users converted.

In [5]:
```
df.query('converted == 1')['user_id'].nunique() / df['user_id'].nunique()
```

Out[5]:
```
0.12104245244060237
```

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

In [6]:
```
df.query('(group == "control" and landing_page == "new_page") or (group == "tr
eatment" and landing_page == "old_page")')['user_id'].count()
```

Out[6]:
```
3893
```

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
```

No missing values

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.

In [8]:
```
dft = df.query('not((group == "control" and landing_page == "new_page") or (gr
oup == "treatment" and landing_page == "old_page"))')
df2 = dft.copy()
```

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[0]
```

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

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

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

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

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

In [11]:
```
df2['user_id'].value_counts().head()
```

Out[11]:
```
773192    2
630732    1
811737    1
797392    1
795345    1
Name: user_id, dtype: int64
```

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

In [12]:
```
df2.query('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**, but keep your dataframe as **df2**.

In [13]:
```
df2.drop(2893, axis=0, inplace=True)

df2.query('user_id == 773192')
```

Out[13]:

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

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

In [14]:
```
df2['converted'].mean()
```

Out[14]:
```
0.11959708724499628
```

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
```

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

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

In [17]:
```
df2.query('landing_page == "new_page"').shape[0] / df2.shape[0]
```

Out[17]: 0.5000619442226688

There doesn't seem to be clear evidence that one page leads to more conversions. The difference between probability of being converted between the 2 pages is a mere 0.0015 which is not sufficient enough for a conclusion.

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.

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?

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

$$H_1 : p_{new} - p_{old} > 0$$

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.

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

In [18]:
```
df2['converted'].mean()
```

Out[18]: 0.11959708724499628

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

In [19]:
```
df2['converted'].mean()
```

Out[19]: 0.11959708724499628

What is n_{new}?

In [20]:
```
df2.query('group == "treatment"')['user_id'].count()
```

Out[20]: 145310

What is n_{old}?

In [21]:
```
df2.query('group == "control"')['user_id'].count()
```

Out[21]: 145274

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**.

In [22]:
```
new_page_converted = np.random.choice([0,1], size=145310, p=[1-df2['converted'
].mean(),df2['converted'].mean()])
new_page_converted.mean()
```

Out[22]: 0.11867731057738627

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**.

In [23]:
```
old_page_converted = np.random.choice([0,1], size=145274, p=[1-df2['converted'
].mean(),df2['converted'].mean()])
old_page_converted.mean()
```

Out[23]: 0.11903712983741069

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

In [24]:
```
new_page_converted.mean() - old_page_converted.mean()
```

Out[24]: -0.00035981926002441 6

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 **p_diffs**.

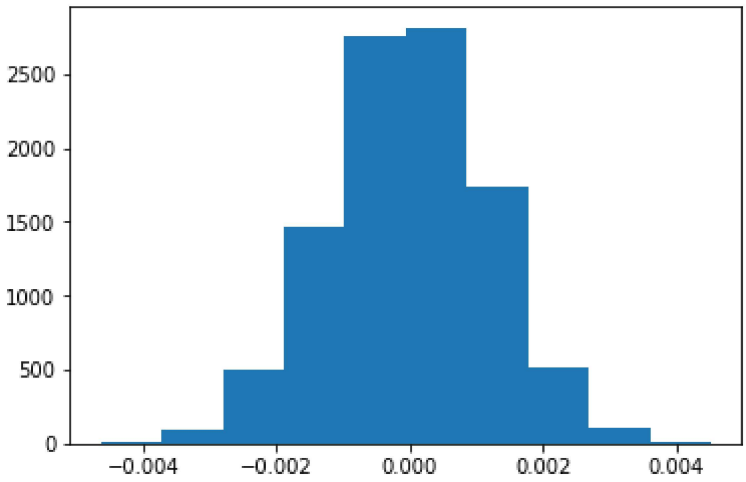
In [25]:
```
p_diffs = []
for n in range(10000):
    new = np.random.choice([0,1], size=145310, p=[1-df2['converted'].mean(),df
2['converted'].mean()])
    old = np.random.choice([0,1], size=145274, p=[1-df2['converted'].mean(),df
2['converted'].mean()])
    p_diffs.append(new.mean()-old.mean())
```

In [26]:
```
p_diffs = np.array(p_diffs)
p_diffs.mean()
```

Out[26]: -8.2078395989945372e-06

Plot a histogram of the **p_diffs**.

In [27]:
```
plt.hist(p_diffs);
```



This is what I expected. In many scenarious we would get a distribution which we would then have to use it's standard deviation to create into a normal distribution, but in this case we already have a normal distribution about the mean 0.

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

In [35]:
```
#actual difference observed in ab_data.csv
p_new = df2.query('landing_page == "new_page"')['converted'].mean()
p_old = df2.query('landing_page == "old_page"')['converted'].mean()
p_act = p_new - p_old
(p_diffs > p_act).mean()
```

Out[35]: 0.90339999999999998

This is (1-p) for our test, which results in a p-value of 0.097. Our Type 1 error is 0.05, Since

$$p > \alpha$$

we accept the null hypothesis.
Since it is stated to "assume under the null hypothesis p_{new} and p_{old} are equal" , we can conclude that there is no difference between the new and the old pages at 5% error.

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 [36]:
```
import statsmodels.api as sm

convert_old = df2.query('group == "control"')['converted'].sum()
convert_new = df2.query('group == "treatment"')['converted'].sum()
n_old = df2.query('group == "control"')['user_id'].count()
n_new = df2.query('group == "treatment"')['user_id'].count()
```

Use `stats.proportions_ztest` to compute your test statistic and p-value. Here (http://knowledgetack.com/python/statsmodels/proportions_ztest/) is a helpful link on using the built in.

In [37]:
```
z_score, p_value = sm.stats.proportions_ztest([convert_old, convert_new], [n_old, n_new], alternative = 'smaller')
```

In [38]:
```
z_score, p_value
```

Out[38]: (1.3109241984234394, 0.90505831275902449)

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?

The p-value agrees with our earlier findings. This p-value leads us to the same conclusion that:

$$p > \alpha$$

so, we accept the null hypothesis.
Since it is stated to "assume under the null hypothesis p_{new} and p_{old} are equal" , we can conclude that there is no difference between the new and the old pages at 5% error.

Part III - A regression approach

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.

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

Logistic Regression

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 colun 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 [39]:
```
ab_page = pd.get_dummies(df2['group'])
df2[['control','ab_page']] = pd.get_dummies(df2['group'])
df2['intercept'] = 1
df2.drop(['control'], axis = 1, inplace = True)
df2.head()
```

Out[39]:

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

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 [42]:
```
import statsmodels.api as sm

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

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

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

In [43]:
```
results.summary()
```

Out[43]:

Logit Regression Results

Dep. Variable:	**converted**	**No. Observations:**	**290584**
Model:	**Logit**	**Df Residuals:**	**290582**
Method:	**MLE**	**Df Model:**	**1**
Date:	**Sat, 11 Aug 2018**	**Pseudo R-squ.:**	**8.077e-06**
Time:	**04:40:57**	**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

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

The p-value associated with ab_page is the result of a hypothesis test conducted on the slope caused by ab_page(coef) to check if it is possibly the true slope of the model or not. Considering that the true slope of the model is to be b_1 then the hypothesis here states:

$$b_1 = -0.015$$
$$b_1 \neq -0.015$$

Hence the higher the p value the higher the probability of the null hypothesis being accepted. This differs from the value found in Part II since that deals with a completely different hypothesis.

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 want to consider other factors in our regression so that we can test for any other possible factors that can be affecting our results which we may not have considered. Adding too many terms into our models might cause our model to become unstable and give us a model with a lower likelihood of success

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 (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?

In [44]:
```
country = pd.read_csv('countries.csv')
df3 = df2.join(country.set_index('user_id'), on ='user_id')
df3.head()
```

Out[44]:

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

In [45]:
```
df3['country'].value_counts()
```

Out[45]:
```
US    203619
UK     72466
CA     14499
Name: country, dtype: int64
```

In [46]:
```
df3[['CA','UK','US']] = pd.get_dummies(df3['country'])
df3.head()
```

Out[46]:

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

In [47]:
```
logit = sm.Logit(df3['converted'], df3[['intercept', 'UK', 'US']])
results = logit.fit()
results.summary()
```

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

Out[47]:

Logit Regression Results

Dep. Variable:	converted	No. Observations:	290584
Model:	Logit	Df Residuals:	290581
Method:	MLE	Df Model:	2
Date:	Sat, 11 Aug 2018	Pseudo R-squ.:	1.521e-05
Time:	04:41:17	Log-Likelihood:	-1.0639e+05
converged:	True	LL-Null:	-1.0639e+05
		LLR p-value:	0.1984

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	-2.0375	0.026	-78.364	0.000	-2.088	-1.987
UK	0.0507	0.028	1.786	0.074	-0.005	0.106
US	0.0408	0.027	1.518	0.129	-0.012	0.093

We can interpret from the coefficients above that a user from CA is:

- **1.05times less likely to be converted than one from UK**
- **1.04times less likely to be converted as one from US.**

This leads us to believe that country does have a slight effect on conversion, specifically if the person is from CA or not.

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.

In [48]:
```
df3['CA_join_abpage'] = df3['CA']*df3['ab_page']
df3['US_join_abpage'] = df3['US']*df3['ab_page']
df3['UK_join_abpage'] = df3['UK']*df3['ab_page']
df3.head()
```

Out[48]:

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

In [53]:
```
logit = sm.Logit(df3['converted'], df3[['intercept', 'US_join_abpage', 'UK_joi
n_abpage']])
results = logit.fit()
```

```
Optimization terminated successfully.
         Current function value: 0.366117
         Iterations 6
```

In [54]:
```
results.summary()
```

Out[54]:

Logit Regression Results

Dep. Variable:	**converted**	**No. Observations:**	**290584**
Model:	**Logit**	**Df Residuals:**	**290581**
Method:	**MLE**	**Df Model:**	**2**
Date:	**Sat, 11 Aug 2018**	**Pseudo R-squ.:**	**1.082e-05**
Time:	**04:42:45**	**Log-Likelihood:**	**-1.0639e+05**
converged:	**True**	**LL-Null:**	**-1.0639e+05**
		LLR p-value:	**0.3164**

	coef	std err	z	P>|z|	[0.025	0.975]
intercept	**-1.9926**	**0.008**	**-252.910**	**0.000**	**-2.008**	**-1.977**
US_join_abpage	**-0.0144**	**0.012**	**-1.155**	**0.248**	**-0.039**	**0.010**
UK_join_abpage	**0.0112**	**0.018**	**0.626**	**0.532**	**-0.024**	**0.046**

We can interpret from the coefficients above that a user from CA who uses the new page is:

- **1.014times more likely to be converted than one from US who uses the new page**
- **1.011times less likely to be converted as one from US who uses the new page.**

The p-values in this case are also significantly higher than before meaning that they are a better fitf or the model.