Analyze A/B Test Results

Table of Contents

Introduction

A/B tests are very commonly performed by data analysts and data scientists.

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
#Setting the seed
random.seed(42)
```

Now, reading the `ab_data.csv` data. Storing it in `df`.

Reading the dataset and having 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

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]:

```
k=df.groupby(['user_id'])['converted'].mean()
t=pd.DataFrame(k)
t.mean()
```

Out[5]:

```
converted    0.119556
dtype: float64
```

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

In [6]:

```
df[((df['group'] == 'treatment') != (df['landing_page'] == 'new_page')) == True].shape[
0]
```

Out[6]:

```
3893
```

Do any of the rows have missing values?

In [7]:

```
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
```

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.

In [13]:

```
df.drop(df.query("group == 'treatment' and landing_page == 'old_page'").index, inplace=True)
df.drop(df.query("group == 'control' and landing_page == 'new_page'").index, inplace=True)
```

In [14]:

```
df.to_csv('ab_edited.csv', index=False)
```

In [52]:

```
df2 = pd.read_csv('ab_edited.csv')
```

In [16]:

```
# 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[16]:

```
0
```

User_ids in **df2**

In [17]:

```
df2.user_id.nunique()
```

Out[17]:

```
290584
```

User_id repeated in **df2**.

In [18]:

```
df2.user_id.value_counts()
```

Out[18]:

```
773192    2
630732    1
811737    1
797392    1
795345    1
801490    1
799443    1
787157    1
793302    1
817882    1
842446    1
815835    1
805596    1
803549    1
809694    1
807647    1
895712    1
840399    1
836301    1
899810    1
834242    1
936604    1
934557    1
940702    1
938655    1
830144    1
828097    1
832195    1
838348    1
821956    1
         ..
734668    1
736717    1
730574    1
775632    1
771538    1
642451    1
773587    1
783828    1
785877    1
779734    1
781783    1
759256    1
726472    1
748999    1
746950    1
753093    1
751044    1
740803    1
738754    1
744897    1
742848    1
634271    1
632222    1
636316    1
630169    1
650647    1
648598    1
654741    1
```

```
652692    1
630836    1
Name: user_id, Length: 290584, dtype: int64
```

Row information for the repeat **user_id**

In [19]:

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

Out[19]:

	user_id	timestamp	group	landing_page	converted
1876	773192	2017-01-09 05:37:58.781806	treatment	new_page	0
2862	773192	2017-01-14 02:55:59.590927	treatment	new_page	0

Removing **one** of the rows with a duplicate **user_id**, but keeping dataframe as **df2**.

In [20]:

```
df2.drop_duplicates('user_id',inplace=True)
```

In [21]:

```
# Checking for above operation
df2.query('user_id == 773192')
```

Out[21]:

	user_id	timestamp	group	landing_page	converted
1876	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 [22]:

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

Out[22]:

```
0.11959708724499628
```

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

In [23]:

```
df_grp = df2.groupby('group')
df_grp.describe()
```

Out[23]:

	converted								user_id	
	count	mean	std	min	25%	50%	75%	max	count	mean
group										
control	145274.0	0.120386	0.325414	0.0	0.0	0.0	0.0	1.0	145274.0	788164.07259
treatment	145310.0	0.118808	0.323564	0.0	0.0	0.0	0.0	1.0	145310.0	787845.71929

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

In [24]:

```
#Answer can inferred from above table for treatment group
```

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

In [25]:

```
df2.query('landing_page == "new_page"').user_id.nunique()/df2.user_id.nunique()
```

Out[25]:

```
0.5000619442226688
```

Conclusion

No, there is insufficient evidence that new treatment leads to more conversions as the results obtained are reverse.

Part II - A/B Test

The **conversion rate** for p_{new} under the null

In [26]:

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

Out[26]:

```
0.11959708724499628
```

The **conversion rate** for p_{old} under the null

In [27]:

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

Out[27]:

```
0.11959708724499628
```

n_{new}, the number of individuals in the treatment group

In [28]:

```
n_new=df2.query('group =="treatment"').user_id.nunique()
n_new
```

Out[28]:

```
145310
```

n_{old}, the number of individuals in the control group

In [29]:

```
n_old=df2.query('group =="control"').user_id.nunique()
n_old
```

Out[29]:

```
145274
```

Simulating n_{new} transactions with a conversion rate of p_{new} under the null and storing these n_{new} 1's and 0's in **new_page_converted**.

In [30]:

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

Out[30]:

```
array([0, 0, 0, ..., 0, 0, 0])
```

Simulate n_{old} transactions with a conversion rate of p_{old} under the null storing these n_{old} 1's and 0's in **old_page_converted**.

In [31]:

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

Out[31]:

```
array([0, 0, 0, ..., 0, 0, 0])
```

p_{new} - p_{old} simulated values

In [32]:

```
obs_diff=new_page_converted.mean()-old_page_converted.mean()
obs_diff
```

Out[32]:

```
-0.00073152211992370082
```

Creating 10,000 p_{new} - p_{old} values using the same simulation process used above. Storing all 10,000 values in a NumPy array called **p_diffs**.

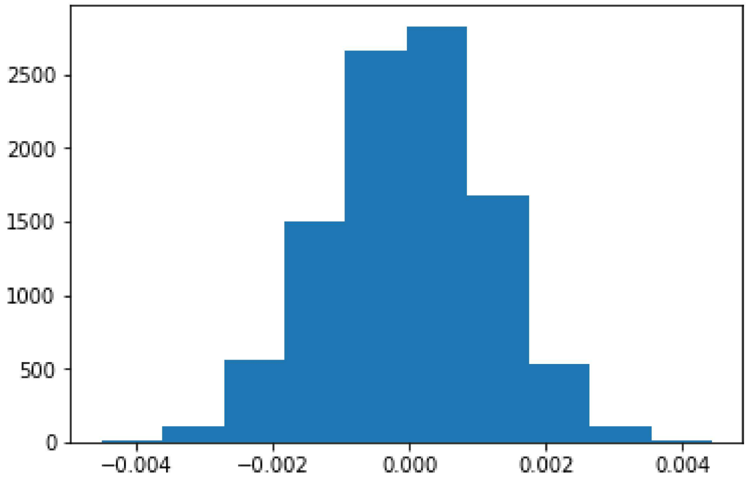
In [33]:

```
p_diffs=[]
size=df.shape[0]
for i in range(10000):
    samp=df2.sample(size,replace=True)
    old_samp_conv=np.random.choice([0,1],n_old, p=(p_old,1-p_old))
    new_samp_conv= np.random.choice([0,1],n_new, p=(p_new,1-p_new))
    p_diffs.append(new_samp_conv.mean()-old_samp_conv.mean())
```

A histogram of the **p_diffs**.This plot looks like what we expected.

In [34]:

```
p_diffs=np.array(p_diffs)
plt.hist(p_diffs)
plt.show()
```



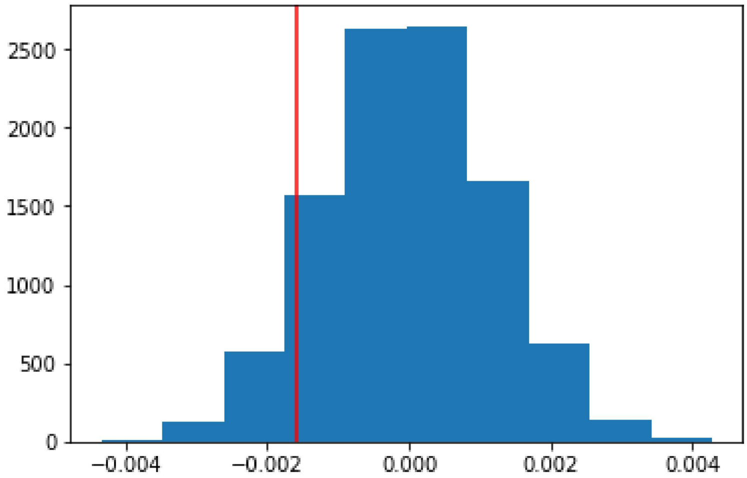
Proportion of the **p_diffs** are greater than the actual difference observed in **ab_data.csv**

In [35]:

```
convert_new = df2.query('converted == 1 and landing_page == "new_page"')['user_id'].nun
ique()
convert_old = df2.query('converted == 1 and landing_page == "old_page"')['user_id'].nun
ique()
actual_cvt_new = float(convert_new)/ float(n_new)
actual_cvt_old = float(convert_old)/ float(n_old)
obs_diff = actual_cvt_new - actual_cvt_old
null_vals = np.random.normal(0, p_diffs.std(), p_diffs.size)
plt.hist(null_vals)
#Plot vertical line for observed statistic
plt.axvline(x=obs_diff,color ='red')
(null_vals > obs_diff).mean()
```

Out[35]:

0.9053



Explanation

Type I error rate of 5%, and Pold > Alpha, we fail to reject the null.

Therefore, the data show, with a type I error rate of 0.05, that the old page has higher probablity of convert rate than new page.

P-Value: The probability of observing our statistic or a more extreme statistic from the null hypothesis.

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.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 number of rows associated with the old page and new pages, respectively.

In [49]:

```
import statsmodels.api as sm

convert_old = df2.query('group == "control"')['converted'].mean()
convert_new = df2.query('group == "treatment"')['converted'].mean()
n_old = df2.query('landing_page == "old_page"').shape[0]
n_new = df2.query('landing_page == "new_page"').shape[0]
```

In [37]:

```
n_old
```

Out[37]:

```
145274
```

Now using `stats.proportions_ztest` to compute our test statistic and p-value.

In [50]:

```
z_score, p_val=sm.stats.proportions_ztest([convert_old, convert_new], [n_old, n_new])
z_score, p_val
```

Out[50]:

```
(0.0032875796753531767, 0.9973768956597913)
```

What do the z-score and p-value computed above mean for the conversion rates of the old and new pages? Do they agree with the findings beforehand.

It indicates that the difference is insignificant.

Hence null hypothesis cannot be rejected which agree with our findings beforehand.

Part III - A regression approach

1. The result we achieved in the A/B test in Part II above can also be achieved by performing regression.

Type of regression we should be performing in this case.

Logistic regression because here we are dealing with categorical variables.

In [53]:

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

Out[53]:

	user_id	timestamp	group	landing_page	converted	intercept	new_page	old_pa
0	851104	2017-01-21 22:11:48.556739	control	old_page	0	1	0.0	1
1	804228	2017-01-12 08:01:45.159739	control	old_page	0	1	0.0	1
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0	1	1.0	0
3	853541	2017-01-08 18:28:03.143765	treatment	new_page	0	1	1.0	0
4	864975	2017-01-21 01:52:26.210827	control	old_page	1	1	0.0	1

Using **statsmodels** to instantiate regression model on the two columns you created in above, then fitting the model using the two columns you created beforehand to predict whether or not an individual converts.

In [54]:

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

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

Summary of the model below

In [55]:

```
results.summary()
```

Out[55]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290585
Model:	Logit	**Df Residuals:**	290583
Method:	MLE	**Df Model:**	1
Date:	Fri, 22 Feb 2019	**Pseudo R-squ.:**	8.085e-06
Time:	12:06:59	**Log-Likelihood:**	-1.0639e+05
converged:	True	**LL-Null:**	-1.0639e+05
		LLR p-value:	0.1897

	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	-1.312	0.190	-0.037	0.007

Now, considering other things that might influence whether or not an individual converts.

We can consider adding new factors such timestamp to decide whether it plays an important role in predicting the results better.

Time stamp can be fuether divided into categories such as morning, Afternoon, Evening etc.

A disadvantage of adding new factors is that it will make the results complex further if the new factors are dependable with existing explanatory variables then we need to add more complex and higher order terms to help predict better results.

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

In [42]:

```
c=pd.read_csv('countries.csv')
c.head()
```

Out[42]:

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

In [56]:

```
df3=df2.merge(c,on='user_id',how='left')
c.country.unique()
```

Out[56]:

```
array(['UK', 'US', 'CA'], dtype=object)
```

In [57]:

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

In [58]:

```
df3=df3.drop(df3['CA'])
df3.head()
```

Out[58]:

	user_id	timestamp	group	landing_page	converted	intercept	new_page	old_
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0	1	1.0	
3	853541	2017-01-08 18:28:03.143765	treatment	new_page	0	1	1.0	
4	864975	2017-01-21 01:52:26.210827	control	old_page	1	1	0.0	
5	936923	2017-01-10 15:20:49.083499	control	old_page	0	1	0.0	
6	679687	2017-01-19 03:26:46.940749	treatment	new_page	1	1	1.0	

In [59]:

```
df3['intercept']=1
```

In [70]:

```
df3['new_page'].value_counts()
```

Out[70]:

```
1.0    143374
0.0    143368
Name: new_page, dtype: int64
```

In [71]:

```
log_mod = sm.Logit(df3['converted'], df3[['intercept','ab_page','UK', 'US']])
results = log_mod.fit()
results.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.366114
         Iterations 6
```

Out[71]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290583
Model:	Logit	**Df Residuals:**	290579
Method:	MLE	**Df Model:**	3
Date:	Fri, 22 Feb 2019	**Pseudo R-squ.:**	2.326e-05
Time:	12:24:39	**Log-Likelihood:**	-1.0639e+05
converged:	True	**LL-Null:**	-1.0639e+05
		LLR p-value:	0.1756

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
intercept	-2.0300	0.027	-76.248	0.000	-2.082	-1.978
ab_page	-0.0150	0.011	-1.309	0.191	-0.037	0.007
UK	0.0408	0.027	1.516	0.129	-0.012	0.093
US	0.0506	0.028	1.784	0.075	-0.005	0.106

From the result above it is clear that the use of columns is not significant in predicting the conversion rate as depicted by the p-values.

In [72]:

```
1/np.exp(-0.015),np.exp(0.0408),np.exp(0.0506)
```

Out[72]:

```
(1.015113064615719, 1.0416437559600236, 1.0519020483004984)
```

Interpreting Result:

For every unit for new_page decrease, convert will be 1.5% more likely to happen, holding all other varible constant.

For every unit for UK increases, convert is 5.2% more to happen, holding all other varible constant.

For every unit for US increases, convert is 4.2% more to happen, holding all other varible constant.

Though we 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.

In [73]:

```
df3['UK_new_page'] = df3['ab_page']* df3['UK']
df3['US_new_page'] = df3['ab_page']* df3['US']
logit4 = sm.Logit(df3['converted'], df3[['intercept','ab_page','UK_new_page','US_new_pa
ge','UK','US']])
result4 = logit4.fit()
result4.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.366110
         Iterations 6
```

Out[73]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290583
Model:	Logit	**Df Residuals:**	290577
Method:	MLE	**Df Model:**	5
Date:	Fri, 22 Feb 2019	**Pseudo R-squ.:**	3.485e-05
Time:	12:30:34	**Log-Likelihood:**	-1.0639e+05
converged:	True	**LL-Null:**	-1.0639e+05
		LLR p-value:	0.1915

	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
UK_new_page	0.0469	0.054	0.871	0.384	-0.059	0.152
US_new_page	0.0783	0.057	1.378	0.168	-0.033	0.190
UK	0.0176	0.038	0.466	0.641	-0.056	0.091
US	0.0118	0.040	0.296	0.767	-0.066	0.090

In [74]:

```
np.exp(result4.params)
```

Out[74]:

```
intercept      0.134794
ab_page        0.934776
UK_new_page    1.047966
US_new_page    1.081428
UK             1.017705
US             1.011854
dtype: float64
```

Interpreting the result

From the above Logit Regression Results, we can see the coefficient of intereaction variable "UK_new_page" and "US_new_page" are different from the coefficient of ab_page itself.

Also,only intercept's p-value is less than 0.05, which is statistically significant enough for converted rate. Other varable in the summary are not statistically significant.

Additionally, Z-score for all X variables are not large enough to be significant for predicting converted rate.

Therefore, the country a user lives is not significant on the converted rate considering the page the user land in.

For every unit for new_page decreases, convert will be 9.34% more likely to happen, holding all other varible constant.

Convert is 1.08 times more likely to happen for US and new page users than CA and new page users, holding all other varible constant.

Convert is 1.04 times more likely to happen for UK and new page users than CA and new page users, holding all other varible constant.

Convert is 1.18 % more likely to happen for the users in US than CA, holding all other varible constant.

Convert is 1.76 % more likely to happen for the users in UK than CA, holding all other varible constant