A/B testing for a landing page - regression with covariates

mahshidxyz (http://www.github.com/mahshidxyz)

July 2020

This study will investigate the results of an A/B test in which a new landing page is tested for an e-commerce website. Unit of diversion is user-id. Conversion was measured for logged in users and each user was supposed to be tested once. The experiment has been run in three countries (CA, UK, US) with different sampling sizes. The experiment duration was about 3 weeks.

I first checked the quality of the data and invariants. User-ids that have experienced both landing pages due to double bucketing were removed. I made sure that control and treatment group sizes were even at both global and country levels. To assess the significance of the observed differences between the conversion rates a z-test for proportion was done. To control for the effect of covariate (country of users) on conversion, a logistic regression model with treatment and country variables was built. Including the covariates in the model may produce a more reliable estimate of the treatment effect, controlling for other factors. The results show that treatment and its interaction with user location does not have a significant effect on the conversion rate (p-value of t-tests > 0.05). User location (specifically US vs non-US) can explain some of the variations in the conversion though (p-value of log likelihood ratio test < 0.05).

Data Import, cleaning, quality check

In [1]:
```
import math
import pandas as pd
import numpy as np
import datetime as dt
import matplotlib.pyplot as plt
import statsmodels.api as sm
```

In [2]:
```
# importing the data
df = pd.read_csv('landing_page_test.csv')
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 294478 entries, 0 to 294477
Data columns (total 6 columns):
 #   Column        Non-Null Count   Dtype 
---  ------        --------------   ----- 
 0   user_id       294478 non-null  int64 
 1   timestamp     294478 non-null  object
 2   country       294478 non-null  object
 3   group         294478 non-null  object
 4   landing_page  294478 non-null  object
 5   converted     294478 non-null  int64 
dtypes: int64(2), object(4)
memory usage: 13.5+ MB
```

In [3]:
```
# quality check
df.group.unique(), df.landing_page.unique()
```

Out[3]:
```
(array(['control', 'treatment'], dtype=object),
 array(['old_page', 'new_page'], dtype=object))
```

In [4]:
```
# quality check
df.country.unique()
```

Out[4]:
```
array(['US', 'CA', 'UK'], dtype=object)
```

In [5]:
```
# quality check for wrong assignments
print(len(df.query("group == 'treatment' and landing_page == 'old_page'")))
print(len(df.query("group == 'control' and landing_page == 'new_page'")))
```

```
1965
1928
```

In [6]:
```
# total conversion count
df.groupby('converted')['user_id'].count()
```

Out[6]:
```
converted
0    256744
1     37734
Name: user_id, dtype: int64
```

In [7]:
```
# number of unique user-ids who converted is less than the total convrsion count
df[df['converted']==1]['user_id'].nunique()
```

Out[7]:
```
37664
```

In [8]:
```
# the duplicates only apear twice in the dataset
df.groupby('user_id').size().sort_values(ascending=False).head()
```

Out[8]:
```
user_id
809993    2
800362    2
800351    2
755787    2
633243    2
dtype: int64
```

In [9]:
```
# I will drop the rows with mistmatched landing page types and group types
df = df[((df['group']=='control') & (df['landing_page']=='old_page')) | ((df['gro
```

In [10]:
```
# new size of dataset
len(df)
```

Out[10]:
```
290585
```

In [11]:
```
# check for duplicated user ids
len(df) - df.user_id.nunique()
```

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

In [12]:
```
# there is still 1 duplicate
df[df.user_id.duplicated(keep=False)]
```

Out[12]:

	user_id	timestamp	country	group	landing_page	converted
1899	773192	2017-01-09 05:37:58.781806	US	treatment	new_page	0
2893	773192	2017-01-14 02:55:59.590927	US	treatment	new_page	0

In [13]:
```
# I will drop this user-id too for consistency
df.drop_duplicates('user_id', inplace=True)
```

Sanity check: Invariants

In [14]:
```
# check if the users are distributed evenly between control and treatment
df.groupby(['country', 'group']).size()
```

Out[14]:
```
country  group
CA       control        7198
         treatment      7301
UK       control       36360
         treatment     36106
US       control      101716
         treatment    101903
dtype: int64
```

In [15]:
```
# check if CA proportion is ok
p = len(df[(df['country']=='CA') & (df['group'] == 'control')])/len(df[df['country
n_CA = len(df[df['country']=='CA'])
# std for a binomial prob of 0.5 with n_CA samples
SD = math.sqrt(0.5*0.5/n_CA)
if p > 0.5 + SD * 1.96 or p < 0.5 - SD * 1.96:
    print ('prob of being in control group in CA is significantly different from (
else:
    print ('We are good! p = {}, 95% CI for 0.5 is [{}, {}]'.format(p, 0.5 - SD *
```

```
We are good! p = 0.49644803089868267, 95% CI for 0.5 is [0.4918612623233678, 0.
5081387376766322]
```

Sanity check: Trends over time

In [16]:
```
df['date'] = pd.to_datetime(df['timestamp']) # returns datetime
df['date'] = df['date'].dt.date
df.head()
```

Out[16]:

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

In [17]:
```
df_time = df.groupby(['date', 'group']).agg({'landing_page': 'count', 'converted'
df_time.rename(columns = {'landing_page':'total'}, inplace = True)
df_time['conversion'] = df_time['converted']/df_time['total']
df_time.head()
```

Out[17]:

	date	group	total	converted	conversion
0	2017-01-02	control	2859	382	0.133613
1	2017-01-02	treatment	2853	362	0.126884
2	2017-01-03	control	6590	805	0.122155
3	2017-01-03	treatment	6618	799	0.120731
4	2017-01-04	control	6578	851	0.129371

In [18]:
```
# no particular entry stands out. we are good.
# first and last day are not full probably since this was run in 3 countries with
df_time.pivot(index='date', columns='group')
```

Out[18]:

	total		converted		conversion	
group	control	treatment	control	treatment	control	treatment
date						
2017-01-02	2859	2853	382	362	0.133613	0.126884
2017-01-03	6590	6618	805	799	0.122155	0.120731
2017-01-04	6578	6541	851	825	0.129371	0.126128
2017-01-05	6427	6505	856	804	0.133188	0.123597
2017-01-06	6606	6747	815	890	0.123373	0.131910
2017-01-07	6604	6609	853	836	0.129164	0.126494
2017-01-08	6687	6700	846	874	0.126514	0.130448
2017-01-09	6628	6615	845	838	0.127489	0.126682
2017-01-10	6654	6696	808	898	0.121431	0.134110
2017-01-11	6688	6673	857	830	0.128140	0.124382
2017-01-12	6522	6637	854	875	0.130941	0.131837
2017-01-13	6552	6508	825	794	0.125916	0.122004
2017-01-14	6548	6599	885	847	0.135156	0.128353
2017-01-15	6714	6549	861	803	0.128239	0.122614
2017-01-16	6591	6545	863	836	0.130936	0.127731
2017-01-17	6617	6538	863	895	0.130422	0.136892
2017-01-18	6482	6603	860	873	0.132675	0.132213
2017-01-19	6578	6552	846	827	0.128611	0.126221
2017-01-20	6534	6679	810	842	0.123967	0.126067
2017-01-21	6749	6560	901	807	0.133501	0.123018
2017-01-22	6596	6669	838	836	0.127047	0.125356
2017-01-23	6716	6633	887	850	0.132073	0.128147
2017-01-24	3754	3681	485	483	0.129196	0.131214

Test result summary

In [19]:
```
# summarize the data
df_summary = df.groupby('group')['converted'].agg({'count','sum','mean'}).reset_i
df_summary.rename(columns = {'count':'n_total', 'sum':'n_converted', 'mean':'conv
df_summary = df_summary[['group', 'n_total', 'n_converted', 'conversion']]
df_summary
```

Out[19]:

	group	n_total	n_converted	conversion
0	control	145274	18696	0.128695
1	treatment	145310	18524	0.127479

A/B test result analysis: z-test for proportion

The difference between the conversion rate in the control and treatment groups looks trivial (0.120 vs 0.118) and the control group actually had a higher conversion rate. To assess the significance of the results I will do a z-test for proportion, in which:

$$p_{pool} = \frac{p_1 n_1 + p_2 n_2}{n_1 + n_2}$$

$$SE = \sqrt{p_{pool}(1 - p_{pool})(\frac{1}{n_1} + \frac{1}{n_2})}$$

$$z = \frac{p_1 - p_2}{SE}$$

Based on this test (one-sided), there is no significant difference between the two groups' conversions.

In [20]:
```
n_control_convert = float(df_summary.loc[df_summary['group']=='control', 'n_conve
n_control_total = float(df_summary.loc[df_summary['group']=='control', 'n_total']
n_treatment_convert = float(df_summary.loc[df_summary['group']=='treatment', 'n_c
n_treatment_total = float(df_summary.loc[df_summary['group']=='treatment', 'n_tot


## 1_sided
z_score, p_value = sm.stats.proportions_ztest([n_control_convert, n_treatment_con
print('Test results for an alternative hypothesis that p-control > p-treatment:')
print('z-score= {}, p-value= {}'.format(z_score, p_value))
print()

## 2-sided for using in a future comparison
z_score, p_value = sm.stats.proportions_ztest([n_control_convert, n_treatment_con
print('Note that if we were interested in a two-sided test (alternative hypothesi
print('z-score= {}, p-value= {}'.format(z_score, p_value))
print('We will see the same p-value of the 2-sided test later in the regression m
```

```
Test results for an alternative hypothesis that p-control > p-treatment:
z-score= 0.9803773714356453, p-value= 0.16344993769848498

Note that if we were interested in a two-sided test (alternative hypothesis p-c
ontrol <> p-treatment), we would have gotton:
z-score= 0.9803773714356453, p-value= 0.32689987539696996
We will see the same p-value of the 2-sided test later in the regression model.
```

A/B test result analysis: Regression

In [21]:
```
# converting categorical variables to binary
df2 = pd.get_dummies(df, columns=['group', 'country'])
df2.head()
```

Out[21]:

	user_id	timestamp	landing_page	converted	date	group_control	group_treatment	count
0	851104	2017-01-21 22:11:48.556739	old_page	0	2017-01-21	1	0	
1	804228	2017-01-12 08:01:45.159739	old_page	0	2017-01-12	1	0	
2	661590	2017-01-11 16:55:06.154213	new_page	0	2017-01-11	0	1	
3	853541	2017-01-08 18:28:03.143765	new_page	0	2017-01-08	0	1	
4	864975	2017-01-21 01:52:26.210827	old_page	1	2017-01-21	1	0	

In [22]:
```
# I will exclude one of them in the regression
# each country column is a linear function of the other two
df2.drop('group_control', axis = 1, inplace = True)
df2.rename(columns={'group_treatment' : 'treatment'}, inplace = True)
df2.head()
```

Out[22]:

	user_id	timestamp	landing_page	converted	date	treatment	country_CA	country_UK	c
0	851104	2017-01-21 22:11:48.556739	old_page	0	2017-01-21	0	0	0	
1	804228	2017-01-12 08:01:45.159739	old_page	0	2017-01-12	0	0	0	
2	661590	2017-01-11 16:55:06.154213	new_page	0	2017-01-11	1	0	0	
3	853541	2017-01-08 18:28:03.143765	new_page	0	2017-01-08	1	0	0	
4	864975	2017-01-21 01:52:26.210827	old_page	1	2017-01-21	0	0	0	

In [23]:
```
# model with treatment as the single variable, note that the LLR p-value is same
model = sm.Logit.from_formula('converted ~ treatment', data = df2).fit()
model.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.382732
         Iterations 6
```

Out[23]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290584
Model:	Logit	**Df Residuals:**	290582
Method:	MLE	**Df Model:**	1
Date:	Tue, 28 Jul 2020	**Pseudo R-squ.:**	4.321e-06
Time:	22:52:13	**Log-Likelihood:**	-1.1122e+05
converged:	True	**LL-Null:**	-1.1122e+05
Covariance Type:	nonrobust	**LLR p-value:**	0.3269

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-1.9125	0.008	-244.102	0.000	-1.928	-1.897
treatment	-0.0109	0.011	-0.980	0.327	-0.033	0.011

In [24]:
```
# adding covariate to the model and interaction terms
# I have added two of three country columns to keep the features linearly indepen
# results: only country-US predictor is significant! p-val = 0.001
model2 = sm.Logit.from_formula('converted ~ treatment + country_UK + country_US +
                               treatment * country_UK + treatment * country_US',
model2.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.382587
         Iterations 6
```

Out[24]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290584
Model:	Logit	**Df Residuals:**	290578
Method:	MLE	**Df Model:**	5
Date:	Tue, 28 Jul 2020	**Pseudo R-squ.:**	0.0003827
Time:	22:52:17	**Log-Likelihood:**	-1.1117e+05
converged:	True	**LL-Null:**	-1.1122e+05
Covariance Type:	nonrobust	**LLR p-value:**	7.103e-17

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-2.0040	0.036	-55.008	0.000	-2.075	-1.933
treatment	-0.0674	0.052	-1.297	0.195	-0.169	0.034
country_UK	0.0118	0.040	0.296	0.767	-0.066	0.090
country_US	0.1249	0.038	3.324	0.001	0.051	0.199
treatment:country_UK	0.0783	0.057	1.378	0.168	-0.033	0.190
treatment:country_US	0.0529	0.054	0.986	0.324	-0.052	0.158

In [25]:
```
# US vs non-US is the our only predictor with significant effect
model3 = sm.Logit.from_formula('converted ~ country_US', data = df2).fit()
model3.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.382598
         Iterations 6
```

Out[25]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290584
Model:	Logit	**Df Residuals:**	290582
Method:	MLE	**Df Model:**	1
Date:	Tue, 28 Jul 2020	**Pseudo R-squ.:**	0.0003541
Time:	22:52:21	**Log-Likelihood:**	-1.1118e+05
converged:	True	**LL-Null:**	-1.1122e+05
Covariance Type:	nonrobust	**LLR p-value:**	7.021e-19

	coef	std err	z	P>|z|	[0.025	0.975]
Intercept	-1.9951	0.010	-190.998	0.000	-2.016	-1.975
country_US	0.1088	0.012	8.822	0.000	0.085	0.133

In [ ]:
```
```