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, user location, and their interaction does not have a significant effect on the conversion rate (p-value of log likelihood ratio tests > 0.05). In other words, variations in these predictors are not able to explain the variation in conversion.

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    259241
1     35237
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]:
```
35173
```

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['group']=='treatment') & (df['landi
```

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']=='CA'])
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 0.5')
else:
    print ('We are good! p = {}, 95% CI for 0.5 is [{}, {}]'.format(p, 0.5 - SD * 1.96, 0.5 + SD * 1.96))
```

```
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': 'sum'}).reset_index()
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	359	0.125568
1	2017-01-02	treatment	2853	342	0.119874
2	2017-01-03	control	6590	750	0.113809
3	2017-01-03	treatment	6618	753	0.113781
4	2017-01-04	control	6578	802	0.121922

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 different time zone
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	359	342	0.125568	0.119874
2017-01-03	6590	6618	750	753	0.113809	0.113781
2017-01-04	6578	6541	802	763	0.121922	0.116649
2017-01-05	6427	6505	792	748	0.123230	0.114988
2017-01-06	6606	6747	762	833	0.115350	0.123462
2017-01-07	6604	6609	799	768	0.120987	0.116205
2017-01-08	6687	6700	795	809	0.118887	0.120746
2017-01-09	6628	6615	793	781	0.119644	0.118065
2017-01-10	6654	6696	751	846	0.112864	0.126344
2017-01-11	6688	6673	795	768	0.118870	0.115091
2017-01-12	6522	6637	796	812	0.122048	0.122344
2017-01-13	6552	6508	766	724	0.116911	0.111248
2017-01-14	6548	6599	830	787	0.126756	0.119260
2017-01-15	6714	6549	809	743	0.120494	0.113452
2017-01-16	6591	6545	803	780	0.121833	0.119175
2017-01-17	6617	6538	813	832	0.122865	0.127256
2017-01-18	6482	6603	809	824	0.124807	0.124792
2017-01-19	6578	6552	789	768	0.119945	0.117216
2017-01-20	6534	6679	753	786	0.115243	0.117682
2017-01-21	6749	6560	850	759	0.125945	0.115701
2017-01-22	6596	6669	786	787	0.119163	0.118009

	total		converted		conversion	
group	control	treatment	control	treatment	control	treatment
date						
2017-01-23	6716	6633	844	803	0.125670	0.121061
2017-01-24	3754	3681	443	448	0.118007	0.121706

Test result summary

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

Out[19]:

	group	n_total	n_converted	conversion
0	control	145274	17489	0.120386
1	treatment	145310	17264	0.118808

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_converted'])
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_converted'])
n_treatment_total = float(df_summary.loc[df_summary['group']=='treatment', 'n_total'])


## 1_sided
z_score, p_value = sm.stats.proportions_ztest([n_control_convert, n_treatment_convert], [n_control_total, n_treat
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_convert], [n_control_total, n_treat
print('Note that if we were interested in a two-sided test (alternative hypothesis p-control <> p-treatment), we
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 model.')
```

```
Test results for an alternative hypothesis that p-control > p-treatment:
z-score= 1.3109241984234394, p-value= 0.09494168724097551

Note that if we were interested in a two-sided test (alternative hypothesis p-control <> p-treatment), we would
have gotton:
z-score= 1.3109241984234394, p-value= 0.18988337448195103
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	country_CA	country_UK	country_US
0	851104	2017-01-21 22:11:48.556739	old_page	0	2017-01-21	1	0	0	0	1
1	804228	2017-01-12 08:01:45.159739	old_page	0	2017-01-12	1	0	0	0	1
2	661590	2017-01-11 16:55:06.154213	new_page	0	2017-01-11	0	1	0	0	1
3	853541	2017-01-08 18:28:03.143765	new_page	0	2017-01-08	0	1	0	0	1
4	864975	2017-01-21 01:52:26.210827	old_page	1	2017-01-21	1	0	0	0	1

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	country_US
0	851104	2017-01-21 22:11:48.556739	old_page	0	2017-01-21	0	0	0	1
1	804228	2017-01-12 08:01:45.159739	old_page	0	2017-01-12	0	0	0	1
2	661590	2017-01-11 16:55:06.154213	new_page	0	2017-01-11	1	0	0	1
3	853541	2017-01-08 18:28:03.143765	new_page	0	2017-01-08	1	0	0	1
4	864975	2017-01-21 01:52:26.210827	old_page	1	2017-01-21	0	0	0	1

In [23]:
```
# model with treatment as the single variable, note that the LLR p-value is same as the p-value of the 2-sided te
model = sm.Logit.from_formula('converted ~ treatment', data = df2).fit()
model.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.366118
         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.:**	8.077e-06
Time:	01:40:34	**Log-Likelihood:**	-1.0639e+05
converged:	True	**LL-Null:**	-1.0639e+05
Covariance Type:	nonrobust	**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
treatment	-0.0150	0.011	-1.311	0.190	-0.037	0.007

In [24]:
```
# adding covariate to the model and interaction terms
# I have added two of three country columns to keep the features linearly independent
model3 = sm.Logit.from_formula('converted ~ treatment + country_UK + country_US + \
                               treatment * country_UK + treatment * country_US', data = df2).fit()
model3.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.366109
         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.:**	3.482e-05
Time:	01:40:36	**Log-Likelihood:**	-1.0639e+05
converged:	True	**LL-Null:**	-1.0639e+05
Covariance Type:	nonrobust	**LLR p-value:**	0.1920

	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	0.0175	0.038	0.465	0.642	-0.056	0.091
treatment:country_UK	0.0783	0.057	1.378	0.168	-0.033	0.190
treatment:country_US	0.0469	0.054	0.872	0.383	-0.059	0.152