Analyze_AB_test_results_notebook

January 22, 2019

0.1 Analyze A/B Test Results

0.2 Table of Contents

0.2.1 Introduction

In this project, we will be working to understand the results of an A/B test run by an e-commerce website. The goal is to help the company understand if they should implement the new page, keep the old page, or perhaps run the experiment longer to make their decision.

Part I - Probability To get started, We'll import necessary libraries.

```
In [169]: import pandas as pd
          import numpy as np
          import random
          import matplotlib.pyplot as plt
          %matplotlib inline
          #We are setting the seed to get the same sequence of numbers
          random.seed(42)
```

- Now, we'll read in the `ab_data.csv data`. Store it in `df`

```
In [170]: # read in the dataset and take a look at the top few rows
          df = pd.read_csv('ab_data.csv')
          df.head()
```

Out[170]:

	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

```
In [171]: df.timestamp.max(), df.timestamp.min()
```

```
Out[171]: ('2017-01-24 13:41:54.460509', '2017-01-02 13:42:05.378582')

In [172]: # find the number of rows in the dataset
          df.shape

Out[172]: (294478, 5)

In [173]: # The number of unique users in the dataset
          df.user_id.nunique()

Out[173]: 290584

In [174]: # The proportion of users converted
          df['converted'].mean()

Out[174]: 0.11965919355605512

In [175]: # The number of times the `new_page` and `treatment` don't line up
          # Either new_page with control OR old_page with treatment
          (df.query('landing_page == "new_page" and group == "control"').shape[0] +
          df.query('landing_page == "old_page" and group == "treatment"').shape[0])

Out[175]: 3893

In [176]: # Check if there are any missing values
          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
```

- Now, we'll handle rows where `treatment` is not aligned with `new_page` or `control` is not aligned with `old_page`

```
In [177]: # Create a new dataset that meets the specifications and Store it in `df2`
          df0 = df.query('landing_page == "old_page" and group != "treatment"')
          df0.shape[0]

Out[177]: 145274

In [178]: df1 = df.query('landing_page == "new_page" and group != "control"')
          df1.shape[0]
```

```
Out[178]: 145311

In [179]: df2 = df0.append(df1)
          df2.head()

Out[179]:    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
          4   864975  2017-01-21 01:52:26.210827  control     old_page          1
          5   936923  2017-01-10 15:20:49.083499  control     old_page          0
          7   719014  2017-01-17 01:48:29.539573  control     old_page          0

In [180]: # 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[180]: 0

In [181]: # Unique `user_id`s are in `df2`
          df2.user_id.nunique()

Out[181]: 290584

In [182]: # Find the `user_id` repeated in `df2`
          df2['user_id'].value_counts()

Out[182]: 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
```

In [183]:
```
# find the row information for the repeat `user_id`
df2[df2['user_id'] == 773192]
```

Out[183]:

	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

In [184]:
```
# Remove one of the rows with a duplicate `user_id`,
# but keeping the dataframe as `df2`
```

```
df2.drop_duplicates(subset=['user_id'], keep='first', inplace=True)
df2[df2['user_id'] == 773192]
```

Out[184]:

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

```
In [185]: # Find the probability of an individual converting
          # regardless of the page they receive
          (df2['converted'] == 1).mean()
```

Out[185]: 0.11959708724499628

```
In [186]: # Given that an individual was in the `control` group,
          # what is the probability they converted?
          control_prob =(df2.query('group == "control"')['converted'] == 1).mean()
          control_prob
```

Out[186]: 0.1203863045004612

```
In [187]: # Given that an individual was in the `treatment` group,
          # what is the probability they converted?
          treat_prob = (df2.query('group == "treatment"')['converted'] == 1).mean()
          treat_prob
```

Out[187]: 0.11880806551510564

```
In [188]: # Find the probability that an individual received the new page
          new = df2[df2['landing_page'] == "new_page"].count().landing_page
          new_prob = new / df2.shape[0]
          new_prob
```

Out[188]: 0.50006194422266881

The results are almost the same and so, it is hard to conclude that one page leads to more conversions or not.

0.2.2 Part II - A/B Test

Notice that because of the time stamp associated with each event, we could technically run a hypothesis test continuously as each observation was observed.

However, The hard question is do we 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 we run to render a decision that neither page is better than another?

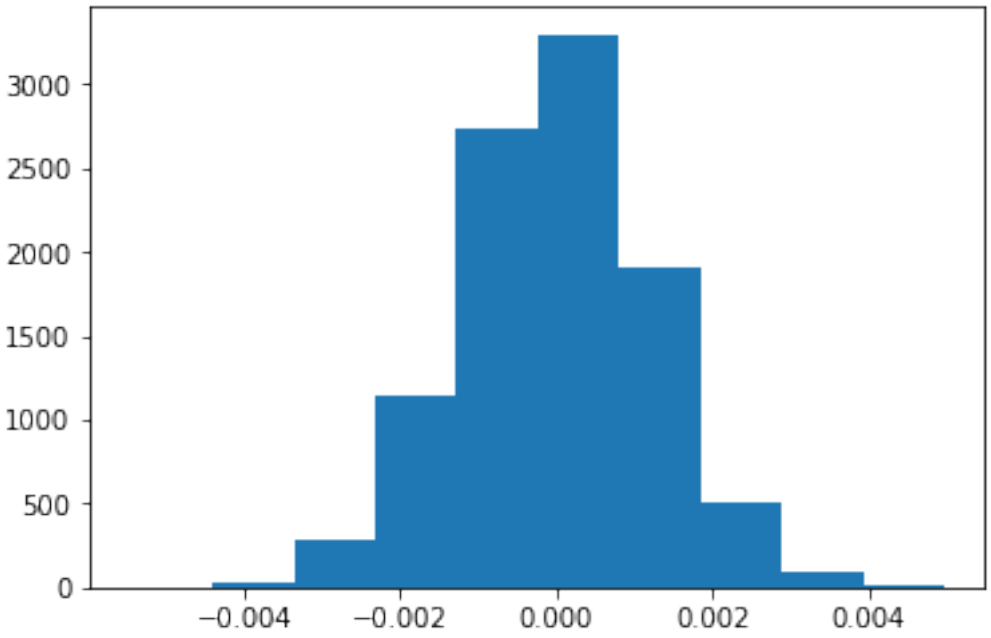
For now, we will assume that the old page is better unless the new page proves to be definitely better at a Type I error rate of 5%. The null and alternative hypotheses will be:

“‘ Null: = ALt: >

“‘

```
In [189]: # Find the `convert rate` for `p_{new}` under the null
          p_new = (df2['converted'] == 1).mean()
          p_new
```

```
Out[189]: 0.11959708724499628

In [190]: # Find the `convert rate` for `p_{old}` under the null
          p_old = (df2['converted'] == 1).mean()
          p_old

Out[190]: 0.11959708724499628

In [191]: # Find `n_{new}`
          n_new = df2[df2['landing_page'] == "new_page"].count().landing_page
          n_new

Out[191]: 145310

In [192]: # Find `n_{old}`
          n_old = df2[df2['landing_page'] == "old_page"].count().landing_page
          n_old

Out[192]: 145274

In [193]: # 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`
          new_page_converted = np.random.choice(2, n_new, p_new)

In [194]: # 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`
          old_page_converted = np.random.choice(2, n_old, p_old)

In [195]: # Find `p_{new}` - `p_{old}` for the simulated values
          diff = new_page_converted.mean() - old_page_converted.mean()
          diff

Out[195]: -0.0029386509123299209

In [196]: # Simulate 10,000 `p_{new}` - `p_{old}` values
          # to the previous parts above
          # Store all 10,000 values in `p_diffs`
          p_diffs = (np.random.binomial(n_new, p_new, 10000)/n_new -
                     np.random.binomial(n_old, p_old, 10000)/n_old)
          p_diffs

Out[196]: array([ 0.00075495,  0.00138123, -0.00130314, ...,  0.00051415,
                  0.00219368, -0.00073151])

In [197]: # Plot a histogram of the `p_diffs`
          plt.hist(p_diffs);
```




```
In [198]: # Find proportion of the `p_diffs` are greater than
          # the actual difference observed in `ab_data.csv`
          new = df2[df2['landing_page'] == "new_page"].count().landing_page
          new_prob = new / df.shape[0]
          new_prob

Out[198]: 0.49344942576355449

In [199]: old = df2[df2['landing_page'] == "old_page"].count().landing_page
          old_prob = old / df.shape[0]
          old_prob

Out[199]: 0.49332717554452288

In [200]: actual_diff = new_prob - old_prob
          actual_diff

Out[200]: 0.00012225021903161659

In [201]: df_diffs = pd.DataFrame(p_diffs)
          df_diffs.columns = ['p_diffs']

In [202]: proportion=len(df_diffs.query('p_diffs > @ actual_diff'))/len(df_diffs)
          proportion

Out[202]: 0.463
```

- We have calculated the number of values (differences) in our simulation greater than the actual difference (probability), this value is the P_Value and it tells us that we fail to reject the null hypothesis and that the new page is not better or the same as the old one

```
In [203]: # calculate the number of conversions for each page,
          # as well as the number of individuals who received each page (using a built-in)
          import statsmodels.api as sm

          convert_old = df2.query('landing_page == "old_page" and converted == 1').shape[0]
          convert_new = df2.query('landing_page == "new_page" and converted == 1').shape[0]
          print(convert_old,convert_new)
          n_old = df2[df2['landing_page'] == "old_page"].count().landing_page
          n_new = df2[df2['landing_page'] == "new_page"].count().landing_page
          print(n_old,n_new)
```

```
17489 17264
145274 145310
```

```
In [204]: # use `stats.proportions_ztest` to compute test statistic and p-value
          z_score, p_value = (sm.stats.proportions_ztest([convert_old, convert_new],
                                                         [n_old, n_new]))
          z_score, p_value
```

```
Out[204]: (1.3109241984234394, 0.18988337448195103)
```

- The z-score and p-value both are greater than 0.05 which mean that we fail to reject the null hypothesis (Z score is between -1.96 and +1.96 and so the p-value is larger than 0.05, we cannot reject the null hypothesis), which means that conversion rates are not high and there is no difference between the old page and the new one. They agree with findings in parts j and k

0.2.3 Part III - A regression approach

Since the values are categorical variables we need to use the logistic regression

```
In [205]: # Create a column for the intercept
          df2['intercept'] = 1
          # Create `ab_page` column, which is 1 when an individual
          # receives the `treatment` and 0 if `control`
          df2['ab_page'] = (df2['group'] == "treatment").astype(int)
          df2.head()
```

```
Out[205]:    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
          4   864975  2017-01-21 01:52:26.210827  control     old_page          1
          5   936923  2017-01-10 15:20:49.083499  control     old_page          0
          7   719014  2017-01-17 01:48:29.539573  control     old_page          0
```

```
   intercept  ab_page
0          1        0
1          1        0
4          1        0
5          1        0
7          1        0
```

In [206]:
```
# Import the regression model using statsmodels
import statsmodels.api as sm
# Instantiate the model
model = sm.Logit(df2['converted'], df2[['intercept','ab_page']])
# fit the model
result = model.fit()
result.summary()
```

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

Out[206]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              converted   No. Observations:               290584
Model:                          Logit   Df Residuals:                   290582
Method:                           MLE   Df Model:                            1
Date:                Tue, 22 Jan 2019   Pseudo R-squ.:               8.077e-06
Time:                        15:22:27   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
==============================================================================
"""
```

- We can see from the above summary that the intercept = -1.9888 and the slope = -0.0150. The p-value of the ab_page is larger than the critical value we would reject the null, in this case, we can say that the specific page does not affect the change in the dependent variable (the coefficient for the specific country is not significant)

- p-value for ab_page = 0.190, in part I I the null and the alternative associated with the same conversion rate regardless of the type of page, in the regression model we see the conversion based on which page a customer receives

- Other factors can help in predicting what may affect our response variable, and in trying to find the best model to predict it. No disadvantages came from the additional terms, they provide more evidence relating to our hypothesis

```
In [207]: # read in the countries.csv dataset
          df3 = pd.read_csv('countries.csv')
          df3.head()
```

```
Out[207]:    user_id country
          0   834778      UK
          1   928468      US
          2   822059      UK
          3   711597      UK
          4   710616      UK
```

```
In [208]: # merge together the two datasets on the approporiate rows
          df4 = df3.set_index('user_id').join(df2.set_index('user_id'), how='inner')
          df4.head()
```

```
Out[208]:         country                   timestamp      group landing_page  \
          user_id
          834778       UK  2017-01-14 23:08:43.304998    control     old_page
          928468       US  2017-01-23 14:44:16.387854  treatment     new_page
          822059       UK  2017-01-16 14:04:14.719771  treatment     new_page
          711597       UK  2017-01-22 03:14:24.763511    control     old_page
          710616       UK  2017-01-16 13:14:44.000513  treatment     new_page

                   converted  intercept  ab_page
          user_id
          834778           0          1        0
          928468           0          1        1
          822059           1          1        1
          711597           0          1        0
          710616           0          1        1
```

```
In [209]: df4.country.unique()
```

```
Out[209]: array(['UK', 'US', 'CA'], dtype=object)
```

```
In [210]: # create dummy variables or the country columns
          df4[['UK', 'US', 'CA']] = pd.get_dummies(df4['country'])
          df4.head()
```

```
Out[210]:         country                   timestamp      group landing_page  \
          user_id
          834778       UK  2017-01-14 23:08:43.304998    control     old_page
          928468       US  2017-01-23 14:44:16.387854  treatment     new_page
          822059       UK  2017-01-16 14:04:14.719771  treatment     new_page
          711597       UK  2017-01-22 03:14:24.763511    control     old_page
```

```
710616       UK  2017-01-16 13:14:44.000513  treatment     new_page

         converted  intercept  ab_page  UK  US  CA
user_id
834778           0          1        0   0   1   0
928468           0          1        1   0   0   1
822059           1          1        1   0   1   0
711597           0          1        0   0   1   0
710616           0          1        1   0   1   0
```

In [211]:
```
# Delete one column of the three dummy variables
df4 = df4.drop('CA', axis = 1)
df4.head()
```

Out[211]:
```
        country                   timestamp      group landing_page  \
user_id
834778       UK  2017-01-14 23:08:43.304998    control     old_page
928468       US  2017-01-23 14:44:16.387854  treatment     new_page
822059       UK  2017-01-16 14:04:14.719771  treatment     new_page
711597       UK  2017-01-22 03:14:24.763511    control     old_page
710616       UK  2017-01-16 13:14:44.000513  treatment     new_page

         converted  intercept  ab_page  UK  US
user_id
834778           0          1        0   0   1
928468           0          1        1   0   0
822059           1          1        1   0   1
711597           0          1        0   0   1
710616           0          1        1   0   1
```

In [212]:
```
# fit the regression model
model2 = sm.Logit(df4['converted'], df4[['intercept','UK', 'US','ab_page']])
result2 = model2.fit()
result2.summary()
```

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

Out[212]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              converted   No. Observations:               290584
Model:                          Logit   Df Residuals:                   290580
Method:                           MLE   Df Model:                            3
Date:                Tue, 22 Jan 2019   Pseudo R-squ.:               2.323e-05
Time:                        15:22:28   Log-Likelihood:            -1.0639e+05
```

```
converged:                       True   LL-Null:                   -1.0639e+05
                                        LLR p-value:                    0.1760
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -1.9893      0.009   -223.763      0.000      -2.007      -1.972
UK            -0.0408      0.027     -1.516      0.130      -0.093       0.012
US             0.0099      0.013      0.743      0.457      -0.016       0.036
ab_page       -0.0149      0.011     -1.307      0.191      -0.037       0.007
==============================================================================
"""
```

- We can see that for the US we predict an increase in conversion by 0.0099 holding other variables constant. For every 100% increase in the UK, the predicted decrease in conversion is -0.0408.
- The only p-value that is statistically significant is the intercept and in this case, we can say that a specific country does not affect the change in the conversion**

Now we will look at an interaction between page and country to see if there significant effects on conversion

In [213]:
```
# Create `ab_test_page` column, which is 1 if `new_page` and 0 if `old_page`
df4['ab_test_page'] = (df4['landing_page'] == "new_page").astype(int)
df4.head()
```

Out[213]:

user_id	country	timestamp	group	landing_page	converted	intercept	ab_page	UK	US	ab_test_page
834778	UK	2017-01-14 23:08:43.304998	control	old_page	0	1	0	0	1	0
928468	US	2017-01-23 14:44:16.387854	treatment	new_page	0	1	1	0	0	1
822059	UK	2017-01-16 14:04:14.719771	treatment	new_page	1	1	1	0	1	1
711597	UK	2017-01-22 03:14:24.763511	control	old_page	0	1	0	0	1	0
710616	UK	2017-01-16 13:14:44.000513	treatment	new_page	0	1	1	0	1	1

In [214]:
```
# Delete one column of the three dummy variables
#df4 = df4.drop('CA', axis = 1)
df4.head()
```

Out[214]:

user_id	country	timestamp	group	landing_page
834778	UK	2017-01-14 23:08:43.304998	control	old_page
928468	US	2017-01-23 14:44:16.387854	treatment	new_page

```
822059       UK  2017-01-16 14:04:14.719771  treatment     new_page
711597       UK  2017-01-22 03:14:24.763511    control     old_page
710616       UK  2017-01-16 13:14:44.000513  treatment     new_page

         converted  intercept  ab_page  UK  US  ab_test_page
user_id
834778           0          1        0   0   1             0
928468           0          1        1   0   0             1
822059           1          1        1   0   1             1
711597           0          1        0   0   1             0
710616           0          1        1   0   1             1
```

In [215]:
```
# fit the regression model
model3 = sm.Logit(df4['ab_test_page'], df4[['intercept','UK', 'US']])
result3 = model3.fit()
result3.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.693144
         Iterations 3
```

Out[215]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:           ab_test_page   No. Observations:               290584
Model:                          Logit   Df Residuals:                   290581
Method:                           MLE   Df Model:                            2
Date:                Tue, 22 Jan 2019   Pseudo R-squ.:               4.442e-06
Time:                        15:22:29   Log-Likelihood:            -2.0142e+05
converged:                       True   LL-Null:                   -2.0142e+05
                                        LLR p-value:                    0.4088
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      0.0018      0.004      0.414      0.679      -0.007       0.011
UK             0.0124      0.017      0.720      0.472      -0.021       0.046
US            -0.0088      0.009     -1.023      0.306      -0.026       0.008
==============================================================================
"""
```

- We can see that for the UK we predict an increase in conversion by 0.0124 holding other variables constant. For every 100% increase in the US, the predicted decrease in conversion is -0.0088
- p-values are not statistically significant, and the coefficient for the specific country is not significant

0.2.4 Conclusions

- Considering all these factors along with the timestamp for this experiment, we can conclude that the duration of this experiment is not enough to decide whether the new page is better or not. Country and Page factors do not affect the change in the conversion rate and the coefficients are not significant and so, we fail to reject the null hypothesis. The company needs to run the experiment longer to make their decision.