Analyze_ab_test_results_notebook

July 24, 2021

0.1 Analyze A/B Test Results

0.2 Table of Contents

Introduction In this project, I will be working on an A/B test run by an e-commerce website. My goal is to help the company to 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

```
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 set up
        random.seed(42)
```

1.

a. Read in the dataset and take 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
```

b. Use the cell below to find the number of rows in the dataset.

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

```
Out[3]: (294478, 5)
```

c. The number of unique users in the dataset.

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

Out[4]: 290584
```

d. The proportion of users converted.

```
In [5]: df['converted'].sum() / unique_users

Out[5]: 0.12126269856564711
```

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

```
In [6]: treat_oldpage = df.query("group == 'treatment' and landing_page != 'new_page'").count()[
        cont_newpage = df.query("group == 'control' and landing_page == 'new_page'").count()[0]
        (treat_oldpage + cont_newpage)

Out[6]: 3893
```

f. 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
```

No there is no missing values.

2.

a. Now use the answer to the quiz to create a new dataset that meets the specifications from the quiz. Store your new dataframe in **df2**.

```
In [8]: dftest = df.drop(df.query("group == 'treatment' and landing_page != 'new_page'").index)
        df2 = dftest.drop(df.query("group == 'control' and landing_page != 'old_page'").index)

In [ ]:
```

```
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].sha

Out[9]: 0
```

3.

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

```
In [10]: df2.shape

Out[10]: (290585, 5)

In [11]: unique_users_df2 = df2['user_id'].nunique()
         unique_users_df2

Out[11]: 290584
```

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

```
In [12]: df2[df2['user_id'].duplicated()]

Out[12]:       user_id                   timestamp      group landing_page  converted
         2893   773192  2017-01-14 02:55:59.590927  treatment     new_page          0
```

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

```
In [13]: 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
         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 [14]: df2 = df2.drop(1899)

In [15]: df2['user_id'].nunique()

Out[15]: 290584

In [16]: df2.shape

Out[16]: (290584, 5)
```

4.

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

```
In [17]: df2['converted'].mean()

Out[17]: 0.11959708724499628
```

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

```
In [18]: (df2.query("group == 'control'")['converted']).mean()
```

```
Out[18]: 0.1203863045004612
```

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

```
In [19]: (df2.query("group == 'treatment'")['converted']).mean()
```

```
Out[19]: 0.11880806551510564
```

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

```
In [20]: df2.query('landing_page == "new_page"').size / df2.size
```

```
Out[20]: 0.50006194422266881
```

e. Consider your results from parts (a) through (d) above, and explain below whether you think there is sufficient evidence to conclude that the new treatment page leads to more conversions.

The new treatment page actually leads to converions less than the old control page, although the diffrence is very small.so there is no sufficient evidence to conclude that the new treatment page leads to more conversions.

Part II - A/B Test

1. The null and alternative hypotheses:

H0: >=

H1: >

2.

a. What is the **conversion rate** for p_{new} under the null?

```
In [30]: Conv_rate = df2['converted'].mean()
         Conv_rate
```

```
Out[30]: 0.11959708724499628
```

b. What is the **conversion rate** for p_{old} under the null?

```
In [31]: Conv_rate = df2['converted'].mean()
         Conv_rate
```

```
Out[31]: 0.11959708724499628
```

c. What is n_{new}, the number of individuals in the treatment group?

```
In [32]: Num_New = df2.query("group == 'treatment'").count()[0]
         Num_New
```

```
Out[32]: 145310
```

d. What is n_{old}, the number of individuals in the control group?

```
In [33]: Num_Old = df2.query("group == 'control'").count()[0]
         Num_Old

Out[33]: 145274
```

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

```
In [34]: Simu_New = np.random.binomial(1, Conv_rate, Num_New)
         Simu_New.mean()

Out[34]: 0.12015690592526324
```

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

```
In [35]: Simu_old = np.random.binomial(1, Conv_rate, Num_Old)
         Simu_old.mean()

Out[35]: 0.11862411718545644
```

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

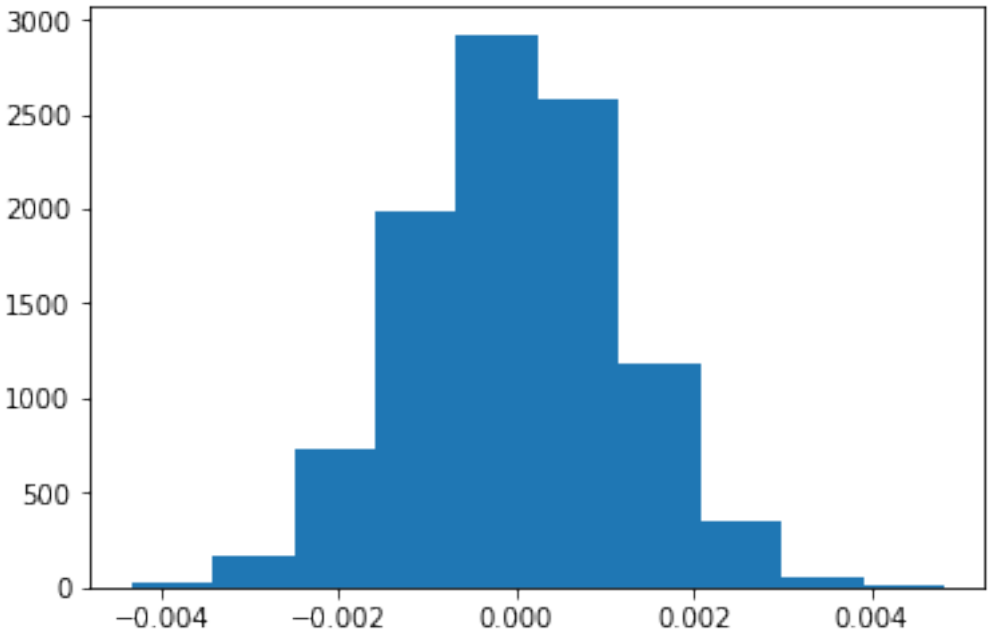
```
In [36]: Simu_New.mean() - Simu_old.mean()

Out[36]: 0.0015327887398067924
```

h. Create 10,000 p_{new} - p_{old} values using the same simulation process you used in parts (a) through (g) above. Store all 10,000 values in a NumPy array called **p_diffs**.

```
In [37]: p_diffs = []
         for _ in range(10000):
             Simu_New = np.random.binomial(1, Conv_rate, Num_New)
             Simu_old = np.random.binomial(1, Conv_rate, Num_Old)
             p_diffs.append(Simu_New.mean() - Simu_old.mean())
```

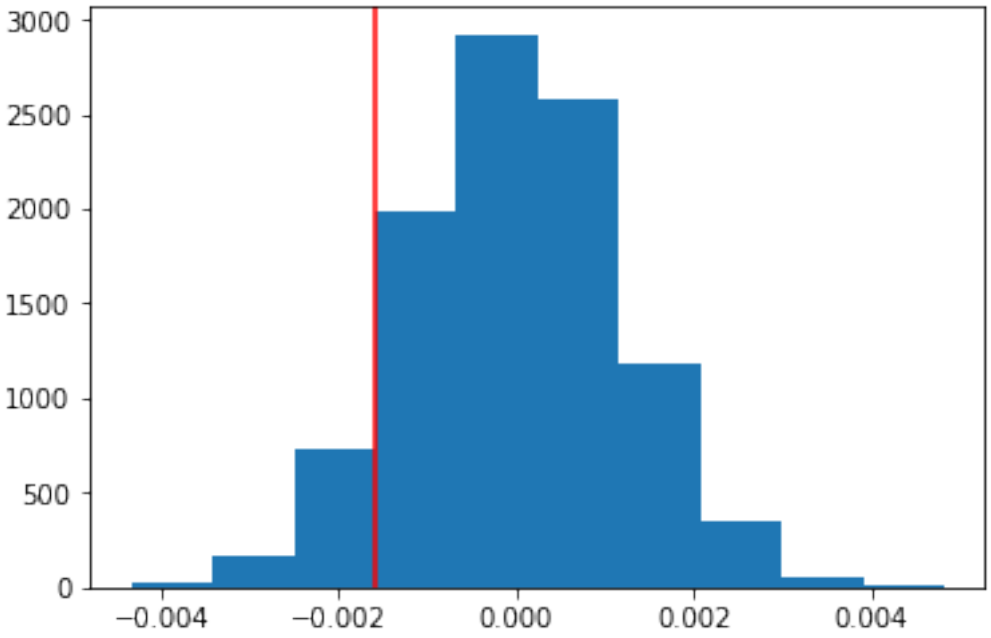
i. Plot a histogram of the **p_diffs**. Does this plot look like what you expected? Use the matching problem in the classroom to assure you fully understand what was computed here.

```
In [38]: plt.hist(p_diffs);
```



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

```
In [39]: act_diff = df2.query('group == "treatment"')['converted'].mean() - df2.query('group ==
         act_diff

Out[39]: -0.0015782389853555567

In [40]: plt.hist(p_diffs);
         plt.axvline(act_diff, c='red')

Out[40]: <matplotlib.lines.Line2D at 0x7f7d3b44a240>
```




```
In [41]: P_Value = (p_diffs > act_diff).mean()
         P_Value
```

```
Out[41]: 0.90600000000000003
```

k. Please explain using the vocabulary you've learned in this course what you just computed in part **j.** What is this value called in scientific studies? What does this value mean in terms of whether or not there is a difference between the new and old pages?

I just computed in part j the P-Value and according to the P-Value if it is lower than 0.05 it means that it is statistically significant and we can reject the null and if it is greater than 0.05 it means that it is not statistically significant and we fail to reject the null. As in this case it is 0.90 so it is not statistically significant and we fail to reject the null.

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

         convert_old = len(df2.query("landing_page == 'old_page' and converted == 1"))
         convert_new = len(df2.query("landing_page == 'new_page'and converted == 1"))
         n_old = len(df2.query("landing_page == 'old_page'"))
         n_new = len(df2.query("landing_page == 'new_page'"))
         convert_old, convert_new, n_old, n_new
```

```
/opt/conda/lib/python3.6/site-packages/statsmodels/compat/pandas.py:56: FutureWarning: The panda
  from pandas.core import datetools
```

```
Out[42]: (17489, 17264, 145274, 145310)
```

m. Now use `stats.proportions_ztest` to compute your test statistic and p-value. Here is a helpful link on using the built in.

```
In [43]: zstat, P_value = sm.stats.proportions_ztest([convert_old, convert_new], [n_old, n_new],
         zstat, P_value
```

```
Out[43]: (1.3109241984234394, 0.90505831275902449)
```

n. 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? Do they agree with the findings in parts **j.** and **k.**?

Zscore value means that the number of standard deviations away from the mean is small and the P-value still greater than 0.05 so it agree with the findings in j and k parts as we fail to reject the null.

Part III - A regression approach

1.

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

Both simple and multiple linear regressions are used to predict quantitative response variable But logistic regression is used to predict categorical response variable and to predict only two possible outcomes.

So in this case I think logistic regression is the type of regression I should use.

b. 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 in df2 a column 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 [44]: df2['intercept'] = 1
         df2['ab_page'] = pd.get_dummies(df2['group'])['treatment']
         df2.head()
```

```
Out[44]:    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

            intercept  ab_page
```

```
0          1        0
1          1        0
2          1        1
3          1        1
4          1        0
```

c. Use **statsmodels** to instantiate your regression model on the two columns you created in part b., then fit the model using the two columns you created in part **b.** to predict whether or not an individual converts.

```
In [45]: df2['intercept'] = 1
         lo_reg = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])
         result = lo_reg.fit()

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

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

```
In [46]: result.summary2()

Out[46]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                 Results: Logit
         =================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2021-07-24 16:53 AIC:              212780.3502
         No. Observations:   290584           BIC:              212801.5095
         Df Model:           1                Log-Likelihood:   -1.0639e+05
         Df Residuals:       290582           LL-Null:          -1.0639e+05
         Converged:          1.0000           Scale:            1.0000
         -----------------------------------------------------------------
                       Coef.   Std.Err.      z      P>|z|    [0.025   0.975]
         -----------------------------------------------------------------
         intercept    -1.9888    0.0081  -246.6690  0.0000  -2.0046  -1.9730
         ab_page      -0.0150    0.0114    -1.3109  0.1899  -0.0374   0.0074
         =================================================================

         """
```

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

The p-value associated with ab_page is 0.189. It differ from the value in Part II because in Part II we assumed that the conversion rate of both old and new pages is the same and the null was that the rate of old page is >= the new page and the alternative that the new page rate is > the old page, But in the regression model we calculated the conversion rate of each page alone.

f. 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?

Of course it is a good idea to add other factors because depending on only one factor may cause not showing the whole picture as maybe there is another factor that really affects the result. About the disadvantages I think it can be considering all factors even if it seems not effect at all, because this will cause a lot of confusion and intrepution.

g. Testing if the conversion rate changes based on which country a user lives in.

```
In [47]: country_df = pd.read_csv('countries.csv')
         country_df.head()
```

Out[47]:

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

```
In [48]: df3 = df2.join(country_df.set_index('user_id'), on='user_id')
         df3.head()
```

Out[48]:

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

```
In [49]: countries = pd.get_dummies(df3['country'])
         New_df = df3.join(countries)
         New_df.head()
```

Out[49]:

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

```
   intercept  ab_page country  CA  UK  US
0          1        0      US   0   0   1
1          1        0      US   0   0   1
2          1        1      US   0   0   1
3          1        1      US   0   0   1
4          1        0      US   0   0   1
```

In [50]:
```
New_df['intercept'] = 1
lo_reg2 = sm.Logit(New_df['converted'], New_df[['intercept', 'ab_page', 'CA', 'UK']])
result2 = lo_reg2.fit()
result2.summary2()
```

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

Out[50]:
```
<class 'statsmodels.iolib.summary2.Summary'>
"""
                          Results: Logit
=================================================================
Model:              Logit            No. Iterations:   6.0000
Dependent Variable: converted        Pseudo R-squared: 0.000
Date:               2021-07-24 16:54 AIC:              212781.1253
No. Observations:   290584           BIC:              212823.4439
Df Model:           3                Log-Likelihood:   -1.0639e+05
Df Residuals:       290580           LL-Null:          -1.0639e+05
Converged:          1.0000           Scale:            1.0000
-----------------------------------------------------------------
               Coef.   Std.Err.      z      P>|z|    [0.025   0.975]
-----------------------------------------------------------------
intercept    -1.9893    0.0089  -223.7628  0.0000  -2.0067  -1.9718
ab_page      -0.0149    0.0114    -1.3069  0.1912  -0.0374   0.0075
CA           -0.0408    0.0269    -1.5161  0.1295  -0.0934   0.0119
UK            0.0099    0.0133     0.7433  0.4573  -0.0162   0.0359
=================================================================

"""
```

According to the p-value of both CA and UK they are > 0.05 so it is not statistically significant so we fail to reject the null.

h. The interaction between page and country to see if there significant effects on conversion.

In [51]:
```
New_df['pages_CA'], New_df['pages_UK'], New_df['pages_US'] =  New_df['ab_page']*New_df[
New_df['ab_page']*New_df['UK'], New_df['ab_page']*New_df['US']
New_df.head()
```

Out[51]:
```
   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

   intercept  ab_page country  CA  UK  US  pages_CA  pages_UK  pages_US
0          1        0      US   0   0   1         0         0         0
1          1        0      US   0   0   1         0         0         0
2          1        1      US   0   0   1         0         0         1
3          1        1      US   0   0   1         0         0         1
4          1        0      US   0   0   1         0         0         0
```

In [53]:
```
New_df['intercept']
lo_reg3 = sm.Logit(New_df['converted'], New_df[['intercept', 'ab_page', 'UK', 'US', 'pa
result3 = lo_reg3.fit()
result3.summary2()
```

```
Optimization terminated successfully.
         Current function value: 0.366109
         Iterations 6
```

Out[53]:
```
<class 'statsmodels.iolib.summary2.Summary'>
"""
                          Results: Logit
=================================================================
Model:              Logit            No. Iterations:   6.0000
Dependent Variable: converted        Pseudo R-squared: 0.000
Date:               2021-07-24 16:54 AIC:              212782.6602
No. Observations:   290584           BIC:              212846.1381
Df Model:           5                Log-Likelihood:   -1.0639e+05
Df Residuals:       290578           LL-Null:          -1.0639e+05
Converged:          1.0000           Scale:            1.0000
-----------------------------------------------------------------
               Coef.   Std.Err.     z      P>|z|    [0.025   0.975]
-----------------------------------------------------------------
intercept     -2.0040    0.0364  -55.0077  0.0000  -2.0754  -1.9326
ab_page       -0.0674    0.0520   -1.2967  0.1947  -0.1694   0.0345
UK             0.0118    0.0398    0.2957  0.7674  -0.0663   0.0899
US             0.0175    0.0377    0.4652  0.6418  -0.0563   0.0914
pages_UK       0.0783    0.0568    1.3783  0.1681  -0.0330   0.1896
pages_US       0.0469    0.0538    0.8718  0.3833  -0.0585   0.1523
=================================================================

"""
```

According to the p-values they are greater than 0.05 so it is not statistically significant so we fail to reject the null. The interaction between page and country has no impact on the conversion rate

Conclusion:

The country has no impact on the conversion rate. Also according to all tests performed in the project the old page is more converted than the new page. So my recommendation to the company is to keep the old page until they improve the new page and retest it.

```
In [54]: from subprocess import call
         call(['python', '-m', 'nbconvert', 'Analyze_ab_test_results_notebook.ipynb'])
```

```
Out[54]: 0
```