Analyze_ab_test_results_notebook

June 14, 2021

0.1 Analyze A/B Test Results

0.2 Table of Contents

- Introduction
- Part I - Probability
- Part II - A/B Test
- Part III - Regression

### Introduction

the results of an A/B test run by an e-commerce website. The goal is to work through this notebook 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, let's import our libraries.

[122]:
```
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. Now, read in the `ab_data.csv` data. Store it in `df`. **Use your dataframe to answer the questions in Quiz 1 of the classroom.**

   a. Read in the dataset and take a look at the top few rows here:

[123]:
```
df = pd.read_csv("ab_data.csv")
df.head()
```

[123]:
```
   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 below cell to find the number of rows in the dataset.

```
[124]: total_rows=len(df.axes[0])
       print("Number of Rows: "+str(total_rows))
```

```
Number of Rows: 294478
```

c. The number of unique users in the dataset.

```
[125]: len(df['user_id'].unique())
```

```
[125]: 290584
```

d. The proportion of users converted.

```
[126]: df['converted'].mean()
```

```
[126]: 0.11965919355605512
```

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

```
[127]: ((df["group"]=="treatment") & (df["landing_page"]=="old_page") |␣
        ↪(df["group"]=="control") & (df["landing_page"]=="new_page")).sum()
```

```
[127]: 3893
```

f. Do any of the rows have missing values?

```
[128]: df.isnull().values.any()
```

```
[128]: False
```

2. For the rows where **treatment** is not aligned with **new_page** or **control** is not aligned with **old_page**, we cannot be sure if this row truly received the new or old page.

```
[129]: df2=df.drop(df.query('(group == "treatment" and landing_page != "new_page") or␣
        ↪(group != "treatment" and landing_page == "new_page") ').index)
```

```
[130]: # 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]
```

```
[130]: 0
```

3. Use **df2** and the cells below to answer questions for **Quiz3** in the classroom.

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

```
[131]: len(df2['user_id'].unique())
```

```
[131]: 290584
```

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

```
[132]: df2[df2.duplicated('user_id')]
```

```
[132]:       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**?

```
[133]: df2[df2['user_id']==773192]
```

```
[133]:       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**.

```
[134]: df2.drop(labels=1899, axis=0, inplace=True)
```

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

```
[135]: print("Probability of user an individual converting regardless of the page they␣
        ↪receive:", df2.converted.mean())
```

```
Probability of user an individual converting regardless of the page they
receive: 0.11959708724499628
```

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

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

```
[136]: 0.1203863045004612
```

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

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

```
[137]: 0.11880806551510564
```

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

```
[138]: print("Probability an individual recieved new page:",
           df2.query('landing_page == "new_page"')['landing_page'].count()/df2.
        ↪shape[0])
```

```
Probability an individual recieved new page: 0.5000619442226688
```

The control group has a convert at rate higher than the treatment with very sight difference and the probability an individual recieved a new page is 0.5. Thus,the probabilities between the two groups are not affected by the size of the group

### Part II - A/B Test

`1.` For now, consider you need to make the decision just based on all the data provided. If you want to assume that the old page is better unless the new page proves to be definitely better at a Type I error rate of 5%, what should your null and alternative hypotheses be? You can state your hypothesis in terms of words or in terms of p_{old} and p_{new}, which are the converted rates for the old and new pages.

$$H_0 : p_{new} - p_{old} <= 0$$

$$H_1 : p_{new} - p_{old} > 0$$

`2.` Assume under the null hypothesis, p_{new} and p_{old} both have "true" success rates equal to the **converted** success rate regardless of page - that is p_{new} and p_{old} are equal. Furthermore, assume they are equal to the **converted** rate in **ab_data.csv** regardless of the page.

Perform the sampling distribution for the difference in **converted** between the two pages over 10,000 iterations of calculating an estimate from the null.

Use the cells below to provide the necessary parts of this simulation. If this doesn't make complete sense right now, don't worry - you are going to work through the problems below to complete this problem. You can use **Quiz 5** in the classroom to make sure you are on the right track.

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

[139]:
```
p_new = df2.converted.mean()
p_new
```

[139]:
```
0.11959708724499628
```

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

[140]:
```
p_old = df2.converted.mean()
p_old
```

[140]:
```
0.11959708724499628
```

c. What is n_{new}?

[141]:
```
n_new = (df2['landing_page']=="new_page").sum()
n_new
```

[141]:
```
145310
```

d. What is n_{old}?

[142]:
```
n_old =  (df2['landing_page']=="old_page").sum()
n_old
```

[142]:
```
145274
```

e. 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**.

[143]:

```
new_page_converted = np.random.binomial(n_new,p_new)
```

f. 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**.

[144]:

```
old_page_converted = np.random.binomial(n_old,p_old)
```

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

[145]:

```
obs_diff= (new_page_converted/n_new - old_page_converted/n_old)
obs_diff
```

[145]:

```
0.0007892141881898213
```

h. Simulate 10,000 p_{new} - p_{old} values using this same process similarly to the one you calculated in parts **a. through g.** above. Store all 10,000 values in a numpy array called **p_diffs**.

[146]:

```
p_diffs = []
for i in range(10000):
    p_new1 = np.random.binomial(n_new,p_new)
    p_old1 = np.random.binomial(n_old,p_old)
    p_diffs.append(p_new1/n_new - p_old1/n_old)
p_diffs = np.array(p_diffs)
```

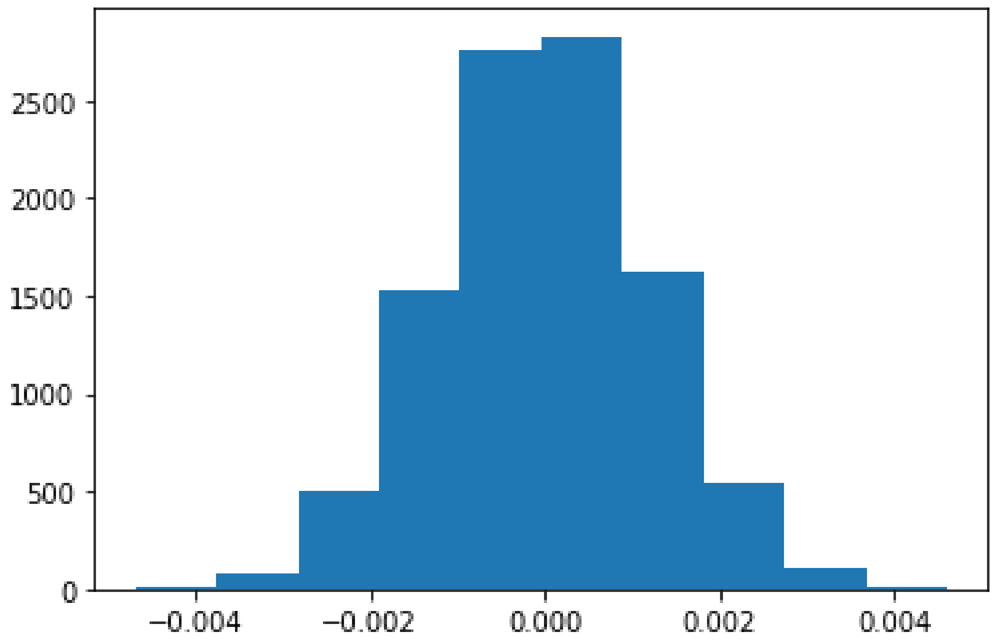
i. Plot a histogram of the **p_diffs**. Does this plot look like what you expected? Use the matching problem in the classroom to assure you fully understand what was computed here.

[147]:

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

[147]:

```
(array([  14.,   83.,  498., 1523., 2759., 2828., 1629.,  541.,  113.,
          12.]),
 array([-4.68222286e-03, -3.75306172e-03, -2.82390059e-03, -1.89473945e-03,
        -9.65578312e-04, -3.64171748e-05,  8.92743963e-04,  1.82190510e-03,
         2.75106624e-03,  3.68022737e-03,  4.60938851e-03]),
 <BarContainer object of 10 artists>)
```



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

[148]:

```
var1 = df2[df2['landing_page'] == 'new_page']['converted'].mean()
var2 = df2[df2['landing_page'] == 'old_page']['converted'].mean()
actual = var1-var2
anwser = ( p_diffs>actual).mean()
anwser
```

[148]:

```
0.9031
```

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

k. In words, explain 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?

This test fails to reject the null hypothesis that there is no difference in the conversion rate of the new and old landing pages since the p-value is more than the appropriate Type 1 error rate (*alpha*) of 0.05,

pval Reject H0

pval > Fail to Reject H0

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.

[149]:

```
import statsmodels.api as sm
convert_old = df2.query('landing_page == "old_page"')['converted'].sum()
convert_new = df2.query('landing_page == "new_page"')['converted'].sum()
n_old = df2.query('landing_page == "old_page"').shape[0]
n_new = df2.query('landing_page == "new_page"').shape[0]
```

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

[150]:

```
z_score, p_value = sm.stats.proportions_ztest([convert_old, convert_new],␣
 ↪[n_old , n_new ],alternative='smaller')
print("The z score: ",z_score)
print("The p value: ",p_value)
```

```
The z score:  1.3109241984234394
The p value:  0.9050583127590245
```

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.**?

The p-value and z-score agree with the results in j and k.Thus, we accept the null hypothesis.

### Part III - A regression approach

`1.` In this final part, you will see that the result you acheived in the previous A/B test can also be acheived by performing regression.

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

Logistic Regression.

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 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**.

[151]:

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

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

[152]:

```
import statsmodels.api as sm
logit_mod = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])
results = logit_mod.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.

[153]:
```
results.summary()
```

[153]:
```
<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:                Mon, 14 Jun 2021   Pseudo R-squ.:               8.077e-06
Time:                        19:29:25   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
ab_page       -0.0150      0.011     -1.311      0.190      -0.037       0.007
==============================================================================
"""
```

e. What is the p-value associated with **ab_page**? Why does it differ from the value you found in **Part II**? **Hint**: What are the null and alternative hypotheses associated with your regression model, and how do they compare to the null and alternative hypotheses in the **Part II**?

The p values in Part II range from 0.9051 to 0.9038. The p value in Part III is 0.19. The discrepancy in p-values is due to the variation in alpha values that we apply in parts II and III.

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?

Another variable is that can be added is time. We may see if the conversion rate changes varies with the time of day. However, Adding extra factor to a regression model has the downside of making the model's analysis more difficult.

g. Now along with testing if the conversion rate changes for different pages, also add an effect based on which country a user lives. You will need to read in the **countries.csv** dataset and merge together your datasets on the approporiate rows. Here are the docs for joining tables.

Does it appear that country had an impact on conversion? Don't forget to create dummy variables for these country columns - **Hint: You will need two columns for the three dummy variables.** Provide the statistical output as well as a written response to answer this question.

```
[154]: dfC = pd.read_csv('countries.csv')
       dfC = df2.merge(dfC, on='user_id', how='inner')
       dfC.head()
```

```
[154]:    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
       0          1        0      US
       1          1        0      US
       2          1        1      US
       3          1        1      US
       4          1        0      US
```

```
[155]: dfC[['CA', 'UK', 'US']] = pd.get_dummies(dfC['country'])
       dfC
```

```
[155]:         user_id                   timestamp      group landing_page  \
       0        851104  2017-01-21 22:11:48.556739    control     old_page
       1        804228  2017-01-12 08:01:45.159739    control     old_page
       2        661590  2017-01-11 16:55:06.154213  treatment     new_page
       3        853541  2017-01-08 18:28:03.143765  treatment     new_page
       4        864975  2017-01-21 01:52:26.210827    control     old_page
       …           …                           …          …           …
       290579   751197  2017-01-03 22:28:38.630509    control     old_page
       290580   945152  2017-01-12 00:51:57.078372    control     old_page
       290581   734608  2017-01-22 11:45:03.439544    control     old_page
       290582   697314  2017-01-15 01:20:28.957438    control     old_page
       290583   715931  2017-01-16 12:40:24.467417  treatment     new_page

               converted  intercept  ab_page country  CA  UK  US
       0               0          1        0      US   0   0   1
       1               0          1        0      US   0   0   1
       2               0          1        1      US   0   0   1
       3               0          1        1      US   0   0   1
       4               1          1        0      US   0   0   1
       …             …          …        …      …   ..  ..  ..
       290579          0          1        0      US   0   0   1
       290580          0          1        0      US   0   0   1
       290581          0          1        0      US   0   0   1
       290582          0          1        0      US   0   0   1
       290583          0          1        1      UK   0   1   0
```

```
[290584 rows x 11 columns]
```

[156]:

```
logit_mod = sm.Logit(dfC['converted'], dfC[['intercept', 'CA', 'UK']])
results = logit_mod.fit()
results.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.366116
         Iterations 6
```

[156]:

```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              converted   No. Observations:               290584
Model:                          Logit   Df Residuals:                   290581
Method:                           MLE   Df Model:                            2
Date:                Mon, 14 Jun 2021   Pseudo R-squ.:               1.521e-05
Time:                        19:29:26   Log-Likelihood:            -1.0639e+05
converged:                       True   LL-Null:                   -1.0639e+05
Covariance Type:            nonrobust   LLR p-value:                    0.1984
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -1.9967      0.007   -292.314      0.000      -2.010      -1.983
CA            -0.0408      0.027     -1.518      0.129      -0.093       0.012
UK             0.0099      0.013      0.746      0.456      -0.016       0.036
==============================================================================
"""
```

[157]:

```
1/np.exp(-0.0408), np.exp(0.0099)
```

[157]:

```
(1.0416437559600236, 1.0099491671175422)
```

CA is less likely to convert, UK is more likely to convert

h. Though you 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. Create the necessary additional columns, and fit the new model.

Provide the summary results, and your conclusions based on the results.

[158]:

```
dfC['UK_new_page'] = dfC['UK']*dfC['ab_page']
dfC['US_new_page'] = dfC['US']*dfC['ab_page']
dfC['CA_new_page'] = dfC['CA']*dfC['ab_page']
dfC.head()
```

[158]:
```
   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  UK_new_page  US_new_page  \
0          1        0      US   0   0   1            0            0
1          1        0      US   0   0   1            0            0
2          1        1      US   0   0   1            0            1
3          1        1      US   0   0   1            0            1
4          1        0      US   0   0   1            0            0

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

[159]:
```
logit_mod = sm.Logit(dfC['converted'], dfC[['intercept','ab_page','CA'␣
 ↪,'US','CA_new_page','US_new_page']])
results = logit_mod.fit()
results.summary()
```

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

[159]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              converted   No. Observations:               290584
Model:                          Logit   Df Residuals:                   290578
Method:                           MLE   Df Model:                            5
Date:                Mon, 14 Jun 2021   Pseudo R-squ.:               3.482e-05
Time:                        19:29:27   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      -1.9922      0.016   -123.457      0.000      -2.024      -1.961
ab_page         0.0108      0.023      0.475      0.635      -0.034       0.056
CA             -0.0118      0.040     -0.296      0.767      -0.090       0.066
```

```
US             0.0057      0.019      0.306      0.760      -0.031       0.043
CA_new_page   -0.0783      0.057     -1.378      0.168      -0.190       0.033
US_new_page   -0.0314      0.027     -1.181      0.238      -0.084       0.021
==============================================================================
"""
```

[161]:

```
1/np.exp(-0.0783), np.exp(-0.0314)
```

[161]:

```
(1.0814470441230692, 0.9690878603945013)
```

CA is less likely to convert, US is more likely to convert

## Conclusions

We can't reject the null hypothesis since the p-value is too high based on simulations, therefore we'll have to maintain the old page.