Analyze_ab_test_results_notebook

April 12, 2020

0.1 Analyze A/B Test Results

You may either submit your notebook through the workspace here, or you may work from your local machine and submit through the next page. Either way assure that your code passes the project RUBRIC. **Please save regularly.**

This project will assure you have mastered the subjects covered in the statistics lessons. The hope is to have this project be as comprehensive of these topics as possible. Good luck!

0.2 Table of Contents

- Section **??**
- Section **??**
- Section **??**
- Section **??**

Introduction

A/B tests are very commonly performed by data analysts and data scientists. It is important that you get some practice working with the difficulties of these

For this project, you will be working to understand the results of an A/B test run by an e-commerce website. Your 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.

As you work through this notebook, follow along in the classroom and answer the corresponding quiz questions associated with each question. The labels for each classroom concept are provided for each question. This will assure you are on the right track as you work through the project, and you can feel more confident in your final submission meeting the criteria. As a final check, assure you meet all the criteria on the RUBRIC.

Part I - Probability

To get started, let's import our libraries.

[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. 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:

[2]:
```
df = pd.read_csv('ab_data.csv')
df.head()
```

[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.

[6]:
```
df.shape[0]
```

[6]:
```
294478
```

c. The number of unique users in the dataset.

[7]:
```
df.user_id.nunique()
```

[7]:
```
290584
```

d. The proportion of users converted.

[8]:
```
df.converted.mean()
```

[8]:
```
0.11965919355605512
```

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

[9]:
```
len(df[(df['group'] != 'treatment') == (df['landing_page'] == 'new_page')])
```

[9]:
```
3893
```

f. Do any of the rows have missing values?

[10]:
```
#df.isnull().values.any()
df.isnull().sum()
```

[10]:
```
user_id         0
timestamp       0
group           0
landing_page    0
converted       0
dtype: int64
```

2. For the rows where **treatment** does not match with **new_page** or **control** does not match with **old_page**, we cannot be sure if this row truly received the new or old page. Use **Quiz 2** in the classroom to figure out how we should handle these rows.

   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**.

[3]:
```
df2 = df.copy()
df2.drop(df2[((df2['group'] == 'treatment') == (df2['landing_page'] ==␣
 ↪'new_page')) == False].index, inplace=True, errors = 'ignore')
```

[4]:
```
# 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]
```

[4]:
```
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**?

[44]:
```
df2.user_id.nunique()
```

[44]:
```
290584
```

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

[5]:
```
df2[df2.user_id.duplicated()]
```

[5]:

	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**?

[8]:
```
df2[df2['user_id'] == 773192]
```

[8]:

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

   d. Remove **one** of the rows with a duplicate **user_id**, but keep your dataframe as **df2**.

[7]:
```
df2.drop_duplicates('user_id', inplace = True)
```

4. Use **df2** in the cells below to answer the quiz questions related to **Quiz 4** in the classroom.

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

[9]:
```
df2.query('converted == 1').shape[0] /  df2.shape[0]
```

[9]:
```
0.11959708724499628
```

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

[42]:
```
control_convert = df2.query('group == "control"')['converted'].mean()
control_convert
```

[42]:
```
0.1203863045004612
```

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

[41]:
```
treatment_convert = df2.query('group == "treatment"')['converted'].mean()
treatment_convert
```

[41]:
```
0.11880806551510564
```

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

[36]:
```
df2.query('landing_page == "new_page"').shape[0] / df2.shape[0]
```

[36]:
```
0.5000619442226688
```

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.

As we can the conversion rate for the control group is higher when compared to the treatment group.

Part II - A/B Test

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

However, then the hard question is do you 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 you run to render a decision that neither page is better than another?

These questions are the difficult parts associated with A/B tests in general.

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.

$$H_0 : p_{new} \leq p_{old}$$

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

OR

$$H_0 : p_{new} - p_{old} \leq 0$$

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

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.

Use a sample size for each page equal to the ones in **ab_data.csv**.

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 **conversion rate** for p_{new} under the null?

[12]:
```
p_new = df2.query('landing_page == "new_page"')['converted'].mean()
p_new
```

[12]:
```
0.11880806551510564
```

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

[13]:
```
p_old = df2.query('landing_page == "old_page"')['converted'].mean()
p_old
```

[13]:
```
0.1203863045004612
```

(GIVEN CONDITION) 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.

[14]:
```
p_mean = (p_old + p_new)/2
p_mean
```

[14]:
```
0.11959718500778342
```

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

[15]:
```
n_new = df2.query('group == "treatment"').shape[0]
n_new
```

[15]:
```
145310
```

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

[16]:
```
n_old = df2.query('group == "control"').shape[0]
n_old
```

[16]:
```
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**.

[34]:
```
new_page_converted = np.random.choice([0,1], size=n_new, p=[(1-p_mean),␣
↪p_mean]).mean()
new_page_converted
```

[34]:
```
0.12003991466519855
```

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**.

[35]:
```
old_page_converted = np.random.choice([0,1], size=n_old, p=[(1-p_mean),␣
↪p_mean]).mean()
old_page_converted
```

[35]:
```
0.11989757286231535
```

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

[36]:
```
p_diff = new_page_converted - old_page_converted
p_diff
```

[36]:
```
0.00014234180288319465
```

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**.

[37]:
```
p_diffs = []
for _ in range(10000):
    new_page_converted = np.random.choice([0,1], size=n_new, p=[(1-p_mean),␣
↪p_mean]).mean()
    old_page_converted = np.random.choice([0,1], size=n_old, p=[(1-p_mean),␣
↪p_mean]).mean()
    p_diffs.append(new_page_converted - old_page_converted)
```

[39]:
```
#conversion of list - p_diffs to array
np.array(p_diffs)
```

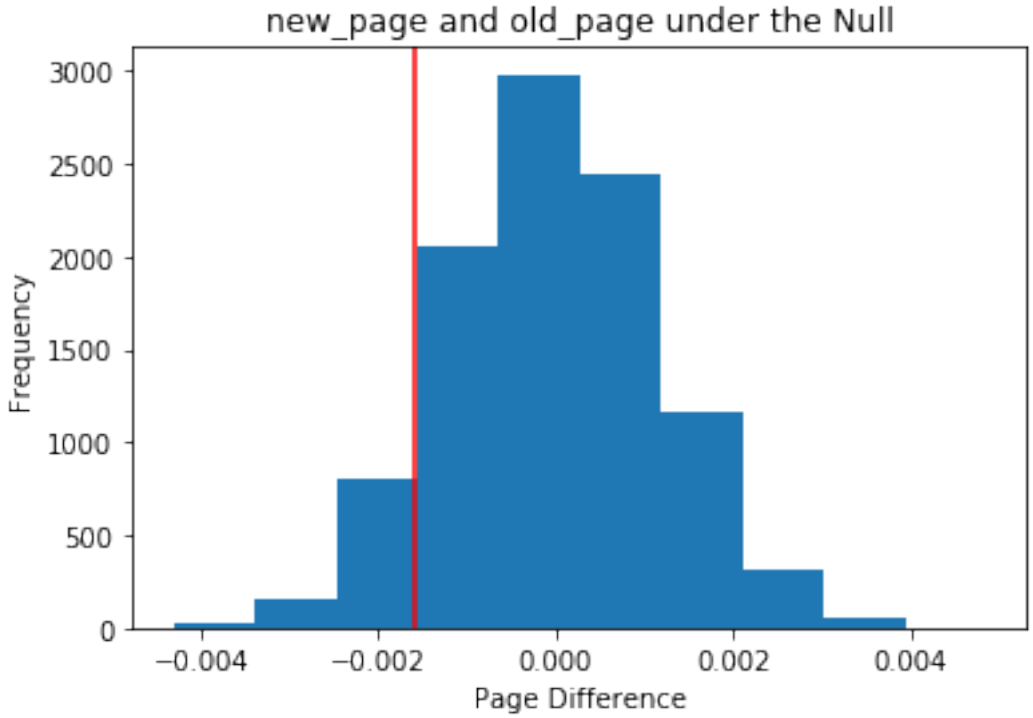
[39]:
```
array([ 0.00030089, -0.00055256,  0.00112673, …, -0.00032542,
        0.0025376 , -0.00104142])
```

[43]:
```
#Actual difference observed in the dataset
obs_diff = treatment_convert - control_convert
obs_diff
```

[43]:
```
-0.0015782389853555567
```

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.

[45]:
```
plt.hist(p_diffs)
plt.title("new_page and old_page under the Null")
plt.xlabel("Page Difference")
plt.ylabel("Frequency")
plt.axvline(obs_diff, c = 'red');
```



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

[46]:
```
(p_diffs > obs_diff).mean()
```

[46]:
```
0.90569999999999995
```

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?

P-value in our observation is 0.9056. This is greater than the type-1 error threshold commonly denoted as α. If α is greater than 0.05 we choose the Null Hypothesis in the business world. Conclusion: We Fail to reject the null hypothesis

$$H_0 : p_{new} - p_{old} \leq 0$$

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.

[143]:
```
import statsmodels.api as sm
sm.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df)

convert_old = df2.query('group == "control" & converted == 1').shape[0]
convert_new = df2.query('group == "treatment" & converted == 1').shape[0]
n_old = df2.query("group == 'control'").shape[0]
n_new = df2.query("group == 'treatment'").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.

[144]:
```
z_score, pval = sm.stats.proportions_ztest([convert_new, convert_old], [n_new,␣
↪n_old], alternative='larger')
z_score,pval
```

[144]:
```
(-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.**?

The z_score value is negative and p-value is larger than our type-1 error threshold. So, we reject the null Hypothesis

Part III - A regression approach

1. In this final part, you will see that the result you achieved in the A/B test in Part II above can also be achieved 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 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**.

[145]:
```
df2['intercept']=1

df2[['control', 'treatment']] = pd.get_dummies(df2['group'])
ab_page = ['treatment', 'control']

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

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.

[146]:
```
import statsmodels.api as sm

logit = sm.Logit(df2['converted'],df2[['intercept','ab_page']])
```

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

[148]:
```
results = logit.fit()
results.summary2()

#had to run summary2 because of statsmodel AttributeError: module 'scipy.stats'␣
 ↪has no attribute 'chisqprob'
#Link : https://www.statsmodels.org/stable/generated/statsmodels.discrete.
 ↪discrete_model.LogitResults.summary2.html?highlight=summary2#statsmodels.
 ↪discrete.discrete_model.LogitResults.summary2
```

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

[148]:
```
<class 'statsmodels.iolib.summary2.Summary'>
"""
                          Results: Logit
=================================================================
Model:              Logit            No. Iterations:   6.0000
Dependent Variable: converted        Pseudo R-squared: 0.000
Date:               2020-04-12 07:16 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**? **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 **Part II**?

p-value = 0.1899

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?

Considering other factors is a good idea as these factors may help to the significance of our test results and leads to more accurate decisions with different results.

One of the most common disadvantages of adding additional variables into the regression model is *Simpson's paradox* where the combined impact of different variables reverses our finding results to what we get when we individually test each variable.

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

[ ]:
```
countries_df = pd.read_csv('countries.csv')
df_new = countries_df.set_index('user_id').join(df2.set_index('user_id'),␣
 ↪how='inner')
df_new.head()
```

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.

[149]:
```
df_new[['US', 'UK']] = pd.get_dummies(df_new['country'])[['US','UK']]
log_mod = sm.Logit(df_new['converted'], df_new[['US', 'UK']])
results = log_mod.fit()
results.summary2()

#had to run summary2 because of statsmodel AttributeError: module 'scipy.stats'␣
 ↪has no attribute 'chisqprob'
```

```
#Link : https://www.statsmodels.org/stable/generated/statsmodels.discrete.
↪discrete_model.LogitResults.summary2.html?highlight=summary2#statsmodels.
↪discrete.discrete_model.LogitResults.summary2
```

```
Optimization terminated successfully.
         Current function value: 0.382864
         Iterations 6
```

[149]:
```
<class 'statsmodels.iolib.summary2.Summary'>
"""
                          Results: Logit
=================================================================
Model:              Logit            No. Iterations:   6.0000
Dependent Variable: converted        Pseudo R-squared: -0.046
Date:               2020-04-12 07:16 AIC:              222512.1606
No. Observations:   290584           BIC:              222533.3199
Df Model:           1                Log-Likelihood:   -1.1125e+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]
-----------------------------------------------------------------
US       -1.9967     0.0068   -292.3145   0.0000   -2.0101   -1.9833
UK       -1.9868     0.0114   -174.1736   0.0000   -2.0092   -1.9645
=================================================================

"""
```

Finishing Up

Congratulations! You have reached the end of the A/B Test Results project! You should be very proud of all you have accomplished!

Tip: Once you are satisfied with your work here, check over your report to make sure that it is satisfies all the areas of the rubric (found on the project submission page at the end of the lesson). You should also probably remove all of the "Tips" like this one so that the presentation is as polished as possible.

0.3 Directions to Submit

Before you submit your project, you need to create a .html or .pdf version of this notebook in the workspace here. To do that, run the code cell below. If it worked correctly, you should get a return code of 0, and you should see the generated .html file in the workspace directory (click on the orange Jupyter icon in the upper left).

Alternatively, you can download this report as .html via the **File** > **Download as** submenu, and then manually upload it into the workspace directory by clicking on the orange Jupyter icon in the upper left, then using the Upload button.

Once you've done this, you can submit your project by clicking on the "Submit Project"

button in the lower right here. This will create and submit a zip file with this .ipynb doc and the .html or .pdf version you created. Congratulations!

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