Analyze_ab_test_results_notebook

April 5, 2019

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.

```
In [2]: 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:

```
In [3]: df = pd.read_csv('ab_data.csv')
        df.head()
```

Out[3]:

	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.

```
In [4]: df.shape
```

```
Out[4]: (294478, 5)
```

c. The number of unique users in the dataset.

```
In [5]: df['user_id'].nunique()
```

```
Out[5]: 290584
```

d. The proportion of users converted.

```
In [6]: df['converted'].mean()
```

```
Out[6]: 0.11965919355605512
```

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

```
In [7]: df1 = df.query('group=="control" and landing_page=="new_page"').shape[0]
        df2 = df.query('group=="treatment" and landing_page=="old_page"').shape[0]

        df1+df2
```

```
Out[7]: 3893
```

f. Do any of the rows have missing values?

```
In [8]: df.isnull().sum()
```

```
Out[8]: user_id         0
        timestamp       0
        group           0
        landing_page    0
        converted       0
        dtype: int64
```

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. Use **Quiz 2** in the classroom to provide 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**.

```
In [9]: df2 =df.query("(group== 'control' and landing_page == 'old_page') or (group== 'treatment
```

```
In [10]: # Double Check all of the correct rows were removed - this should be 0
         df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].sh
```

```
Out[10]: 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**?

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

```
Out[11]: 290584
```

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

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

```
Out[12]: array([773192])
```

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

```
In [13]: df2[df2.duplicated(['user_id'], keep = False)]
```

```
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_duplicates(['user_id'], keep = 'first')
```

4. Use **df2** in the below cells 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?

```
In [15]: df2['converted'].mean()
```

```
Out[15]: 0.11959708724499628
```

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

```
In [16]: df2.head()
```

```
Out[16]:    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 [17]: control_df = df2.query('group == "control"')
         control_df['converted'].mean()
```

```
Out[17]: 0.1203863045004612
```

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

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

```
Out[18]: 0.11880806551510564
```

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

```
In [19]: len(df2.query('landing_page == "new_page"'))/df2.shape[0]
```

```
Out[19]: 0.5000619442226688
```

e. Use the results in the previous two portions of this question to suggest if you think there is evidence that one page leads to more conversions? Write your response below.

Response

I do not think there is clear evidence that one page either old page or new page would lead to more conversions, since conversion results for all dataset, as well as individual results for each group are very close one from the other:

- Probability of an individual converting regardless of the page they receive: 11.96%
- Probability of an individual in the control group to convert: 12.04%
- Probability of an indiviual in the treatment group to convert: 11.88%

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.

Response

Hypotheses

Ho: 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.

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

```
In [20]: p_new = df2['converted'].mean()
         p_new

Out[20]: 0.11959708724499628
```

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

```
In [21]: p_old = df2['converted'].mean()
         p_old

Out[21]: 0.11959708724499628
```

c. What is n_{new}?

```
In [22]: n_new = df2.query('group =="treatment"').shape[0]
         n_new

Out[22]: 145310
```

d. What is n_{old}?

```
In [23]: n_old = df2.query('group =="control"').shape[0]
         n_old

Out[23]: 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**.

```
In [24]: new_page_converted = np.random.binomial(n_new,p_new)
         new_page_converted
```

```
Out[24]: 17370
```

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**.

```
In [25]: old_page_converted = np.random.binomial(n_old,p_old)
         old_page_converted
```

```
Out[25]: 17415
```

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

```
In [26]: new_page_converted/n_new - old_page_converted/n_old
```

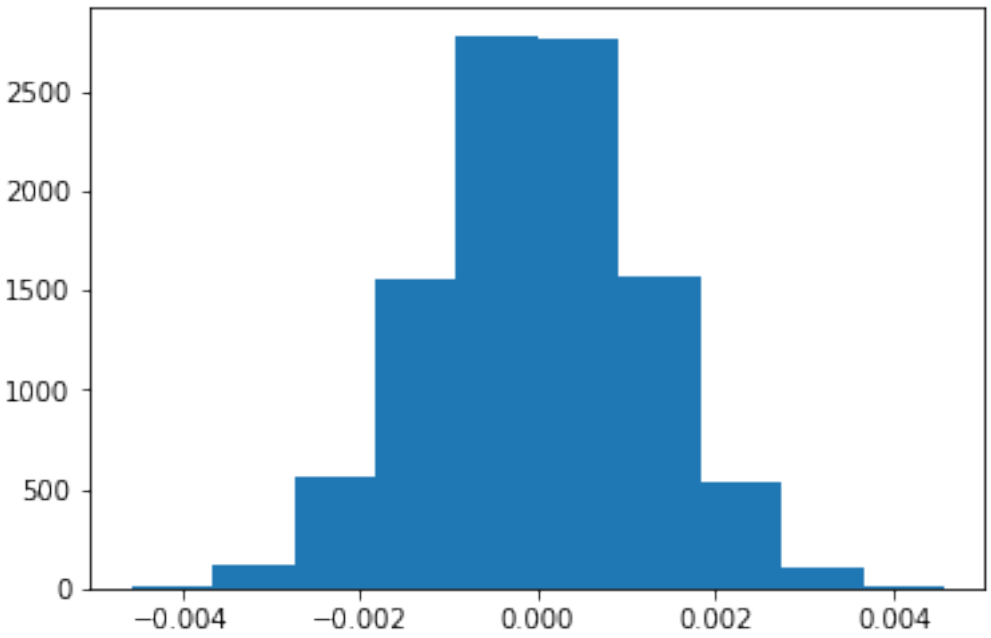
```
Out[26]: -0.0003393817989145176
```

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 **p_diffs**.

```
In [27]: p_diffs = []
         for _ in range(10000):
             new_page_converted = np.random.binomial(n_new,p_new)
             old_page_converted = np.random.binomial(n_old,p_old)
             diff = new_page_converted/n_new - old_page_converted/n_old
             p_diffs.append(diff)
```

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 [28]: plt.hist(p_diffs);
```



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

```
In [29]: control = df2.query('group=="control"')
         conv_control = control['converted'].mean()

         treatment = df2.query('group=="treatment"')
         conv_treatment = treatment['converted'].mean()

         act_diff = conv_treatment - conv_control
         act_diff

Out[29]: -0.0015782389853555567

In [30]: #convert p_diff to numpy array
         p_diff = np.array(p_diffs)

         #Compute p-values
         (act_diff < p_diff).mean()

Out[30]: 0.90280000000000005
```

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?

Response

This value is called: **p-value**

Since p-value is over alpha level of 0.05, this suggests that there is no statistically significant difference with a Type 1 error rate of 0.05, thus we do not have evidence that the new page is better than the old page.

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 [31]: 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]

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

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

```
In [32]: convert_old, convert_new
```

```
Out[32]: (17489, 17264)
```

```
In [33]: n_old, n_new
```

```
Out[33]: (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 [37]: z_score, p_value = sm.stats.proportions_ztest([convert_old, convert_new], [n_old, n_new

         z_score, p_value
```

```
Out[37]: (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.**?

```
In [38]: from scipy.stats import norm

         # Tells us how significant our z-score is
         norm.cdf(z_score)
```

```
Out[38]: 0.90505831275902449
```

```
In [39]: #Tells us what our critical value at 95% confidence is
         norm.ppf(1-(0.05))
```

```
Out[39]: 1.6448536269514722
```

We fail to reject the null hypothesis which suggests that the new page conversion rate is higher than the old page, due that z-score of 1.31 is less than the critical value of 1.64... These finding agree to parts j and k

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 colun 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 [36]: df2['intercept'] = 1
         df2[['control', 'treatment']] = pd.get_dummies(df2['group'])
         df2.head()
```

Out[36]:

	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	control	treatment
0	1	1	0
1	1	1	0
2	1	0	1
3	1	0	1
4	1	1	0

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.

```
In [37]: import statsmodels.api as sm

         logit = sm.Logit(df2['converted'], df2[['intercept', 'treatment']])
         results = logit.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 [38]: results.summary()
```

```
Out[38]: <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:                Sat, 12 Jan 2019   Pseudo R-squ.:               8.077e-06
Time:                        13:50:45   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
treatment     -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 the **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**?

Response

The new P-value associated to ab_page is 0.19, this P-value is different from the one calculated under part II (0.09), since I calculated a one-sided test under Part II and new P-value is assuming a Two-Sided test

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?

Response

The more factors we include, the more likely we would observe significant differences just by chance, we would have to implement the Bonferroni correction to ensure our results are statistically significant, adittionally we could also observe Multicollinearity, due that we could have predictor variables that are correlated with one another, and this could lead to coefficients being flipped from the direction we expect with simple linear regression

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 varaibles.** Provide the statistical output as well as a written response to answer this question.

```
In [39]: countries_df = pd.read_csv('countries.csv')
         countries_df.head()
```

```
Out[39]:    user_id country
         0   834778      UK
         1   928468      US
         2   822059      UK
         3   711597      UK
         4   710616      UK
```

```
In [43]: #Check How many countries are included in dataset and its counts
         countries_df['country'].value_counts()

Out[43]: US    203619
         UK     72466
         CA     14499
         Name: country, dtype: int64

In [44]: #Merging Datasets
         new_df = countries_df.set_index('user_id').join(df2.set_index('user_id'), how = 'inner'
         new_df.head()

Out[44]:         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  control  treatment
         user_id
         834778           0          1        1          0
         928468           0          1        0          1
         822059           1          1        0          1
         711597           0          1        1          0
         710616           0          1        0          1

In [45]: new_df['intercept'] = 1

         #Creating Dummy variables for Country
         new_df[['US', 'UK']] = pd.get_dummies(new_df['country'])[['US', 'UK']]
```

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.

```
In [46]: new_model = sm.Logit(new_df['converted'], new_df[['intercept', 'US', 'UK']])
         results = new_model.fit()
         results.summary()

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

Out[46]:
```
<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:                Sat, 12 Jan 2019   Pseudo R-squ.:               1.521e-05
Time:                        14:02:26   Log-Likelihood:            -1.0639e+05
converged:                       True   LL-Null:                   -1.0639e+05
                                        LLR p-value:                    0.1984
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -2.0375      0.026    -78.364      0.000      -2.088      -1.987
US             0.0408      0.027      1.518      0.129      -0.012       0.093
UK             0.0507      0.028      1.786      0.074      -0.005       0.106
==============================================================================
"""
```

Finishing Up

Congratulations! You have reached the end of the A/B Test Results project! This is the final project in Term 1. You should be very proud of all you have accomplished!

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!

In [40]:
```
from subprocess import call
call(['python', '-m', 'nbconvert', 'Analyze_ab_test_results_notebook.ipynb'])
```

Out[40]:
```
0
```

In [ ]: