Analyze_ab_test_results_notebook

November 9, 2018

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 [38]: 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 [39]:
```
#Load data
df = pd.read_csv('ab_data.csv')

# Show first rows
df.head()
```

Out[39]:

	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 [40]:
```
# Finding out how many rows the data set has
df.shape
```

Out[40]: (294478, 5)

c. The number of unique users in the dataset.

In [41]:
```
# Checking for unique user id via nunique function
df.nunique()
```

Out[41]:
```
user_id         290584
timestamp       294478
group                2
landing_page         2
converted            2
dtype: int64
```

d. The proportion of users converted.

In [42]:
```
# Calculating proportion
df['converted'].mean()
```

Out[42]: 0.11965919355605512

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

In [43]:
```
# Finding number treatment and old page
df_trol = df.query('group == "treatment" and landing_page == "old_page"')
len(df_trol)
# Finding number control and new page
```

```
df_cone = df.query('group == "control" and landing_page == "new_page"')
len(df_cone)
# Sum
total = len(df_trol) + len(df_cone)
print(total)
```

```
3893
```

f. Do any of the rows have missing values?

```
In [44]: # Checking for null values in the complete data set
         df.isnull().sum()
```

```
Out[44]: 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 [45]: # Delete rows
         # First: rows with template and old page
         df.drop(df.query('group == "treatment" and landing_page == "old_page"').index, inplace
         # Second: rows with control and new page
         df.drop(df.query('group == "control" and landing_page == "new_page"').index, inplace =
         # Create df2
         df2 = df.copy()
```

```
In [46]: # 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[46]: 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 [47]: df2.nunique()
```

```
Out[47]: user_id         290584
         timestamp       290585
         group                2
         landing_page         2
         converted            2
         dtype: int64
```

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

```
In [48]: # inspect duplicate userid
         df2[df2.duplicated(['user_id'], keep=False)]['user_id']
```

```
Out[48]: 1899    773192
         2893    773192
         Name: user_id, dtype: int64
```

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

```
In [49]: # Display duplicate rows
         df2[df2.duplicated(['user_id'], keep=False)]
```

Out[49]:

	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 [50]: # Remove the row with timestamp 2017-01-09 05:37:58.781806
         df2 = df2[df2['timestamp'] != '2017-01-09 05:37:58.781806']
         df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 290584 entries, 0 to 294477
Data columns (total 5 columns):
user_id         290584 non-null int64
timestamp       290584 non-null object
group           290584 non-null object
landing_page    290584 non-null object
converted       290584 non-null int64
dtypes: int64(2), object(3)
memory usage: 13.3+ MB
```

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 [51]: # since values are 1 and 0, we can calculate mean to get probability of an individual c
         df['converted'].mean()
```

```
Out[51]: 0.11959667567149027
```

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

```
In [52]: #Probability of a user converted in control group
         df2[df2['group'] == "control"]['converted'].mean()
```

Out[52]: 0.1203863045004612

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

```
In [53]: #Probability of a user converted in treatment group
         df2[df2['group'] == "treatment"]['converted'].mean()
```

Out[53]: 0.11880806551510564

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

```
In [54]: #Probability of a user landing on new_page
         (df2.landing_page == "new_page").mean()
```

Out[54]: 0.50006194422266881

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.

Answer: >According to above proportions, there is a small difference between users converted from treatment group and from control group, and, therefore we cannot conclude that the new treatment page leads to more conversions.

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} - 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 [55]: #Find the proportion of converted rate assuming p_new and p_old are equal
         p_new = df2['converted'].mean()
         p_new
```

```
Out[55]: 0.11959708724499628
```

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

```
In [56]: #Find the proportion of converted rate assuming p_new and p_old are equal
         p_old = df2['converted'].mean()
         p_old
```

```
Out[56]: 0.11959708724499628
```

c. What is n_{new}?

```
In [57]: #Number of users landing on new page
         n_new = df2.query('group == "treatment"')['user_id'].count()
         n_new = int(n_new)
         n_new
```

```
Out[57]: 145310
```

d. What is n_{old}?

```
In [58]: #Number of users landing on old page
         n_old = df2.query('group == "control"')['user_id'].count()
         n_old = int(n_old)
         n_old
```

```
Out[58]: 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 [59]: #Draw samples from a binomial distribution
         new_page_converted = np.random.binomial(1, p_new, n_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**.

```
In [60]: #Draw samples from a binomial distribution
         old_page_converted = np.random.binomial(1, p_old,n_old)
```

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

```
In [61]: #Number of rows from new page are higher than the ones on old page, therefore we trunca
         #page and compute the difference
         new_page_converted = new_page_converted[:145274]
         new_page_converted.mean() - old_page_converted.mean()
```

Out[61]: -0.00078472403871304719

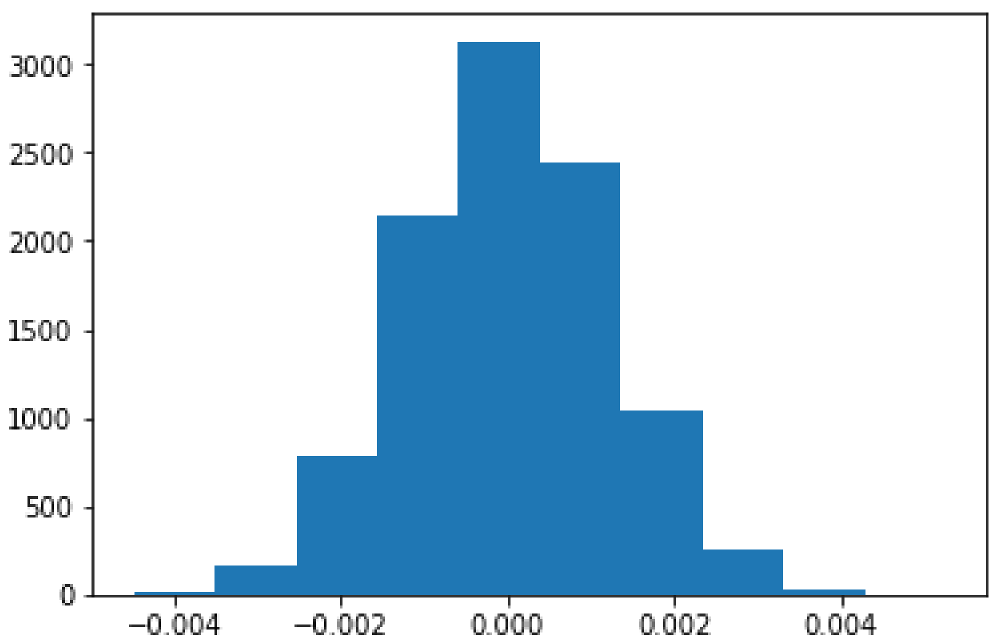
h. Simulate 10,000 p_{new} - p_{old} values using this same process similarly to the one you calculated in parts **a. through g.** above. Store all 10,000 values in **p_diffs**.

```
In [62]: #Simulate 10000 samples of the differences in conversion rates
         p_diffs = []

         for _ in range(10000):
             new_page_converted = np.random.binomial(1, p_new, n_new)
             old_page_converted = np.random.binomial(1, p_old, n_old)
             new_page_p = new_page_converted.mean()
             old_page_p = old_page_converted.mean()
             p_diffs.append(new_page_p - old_page_p)
```

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 [63]: #Show the histogram
         plt.hist(p_diffs);
```



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

```
In [64]: #Actual difference of converted rates
         actual_diff = (df2[df2['group'] == "treatment"]['converted'].mean()) - (df2[df2['group'
         actual_diff
```

Out[64]: -0.0015782389853555567

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?

Answer: >* We are computing p-values here.

- As explained in the videos and quizzes, this is the probability of observing our statistic (or one more extreme in favor of the alternative) if the null hypothesis is true.
- The more extreme in favor of the alternative portion of this statement determines the shading associated with your p-value.
- Here, we find that there is no conversion advantage with new pages. We conclude that null hypothesis is true as old and new pages perform almost similarly. Old pages, as the numbers show, performed slightly better.

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 [65]:
```
import statsmodels.api as sm

#Number of conversions for each page
convert_old = sum(df2.query('group == "control"')['converted'])
convert_new = sum(df2.query('group == "treatment"')['converted'])

#Number of individuals who received each page
n_old = df2.query("group == 'control'")['user_id'].count()
n_new = df2.query("group == 'treatment'")['user_id'].count()

#Convert figures to integers
n_old = int(n_old)
n_new = int(n_new)
```

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 [66]:
```
#Two-sample Proportion Hypothesis Testing
z_score, p_value = sm.stats.proportions_ztest([convert_new, convert_old], [n_new, n_old
z_score
```

Out[66]: -1.3109241984234394

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 [67]: p_value
```

```
Out[67]: 0.90505831275902449
```

Put your answer here.
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 [68]: #Create intercept column
         df2['intercept']=1

         #Create dummies
         ab_page = ['treatment', 'control']
         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.

```
In [69]: 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.

```
In [70]: results = logit.fit()
         results.summary()
```

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

```
Out[70]: <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:                Wed, 31 Oct 2018   Pseudo R-squ.:               8.077e-06
Time:                        18:30:28   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
==============================================================================
"""
```

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**?

The p-value associated with ab_page column is 0.19 which is lower than the p-value calculated using the z-score function. The reason why is different is due to the intercept added.

The logistic regression determines only two possible outcomes. If the new page is equal to the old page or different.

H_0: p_{new} - p_{old} = 0
H_1: p_{new} - p_{old} != 0

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?

We could consider introducing the timestamp metric to determine in which part of the day the individuals converted the most. For example, if we find that the evening is the period that users spend most of their time on the internet we might also take it into consideration.

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

```
In [72]: ### Create the necessary dummy variables
         df_new[['CA', 'US']] = pd.get_dummies(df_new['country'])[['CA','US']]
         df_new.head()
```

```
Out[72]:         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  CA  US
         user_id
         834778           0          1        0   0   0
         928468           0          1        1   0   1
         822059           1          1        1   0   0
         711597           0          1        0   0   0
         710616           0          1        1   0   0
```

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 [73]: ### Fit Your Linear Model And Obtain the Results
         df_new['intercept'] = 1
         log_mod = sm.Logit(df_new['converted'], df_new[['CA', 'US', 'intercept', 'ab_page']])
         results = log_mod.fit()
         results.summary()
```

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

```
Out[73]: <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:                Wed, 31 Oct 2018   Pseudo R-squ.:               2.323e-05
         Time:                        18:30:29   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]
------------------------------------------------------------------------------
CA            -0.0506      0.028     -1.784      0.074      -0.106       0.005
US            -0.0099      0.013     -0.743      0.457      -0.036       0.016
intercept     -1.9794      0.013   -155.415      0.000      -2.004      -1.954
ab_page       -0.0149      0.011     -1.307      0.191      -0.037       0.007
==============================================================================
"""
```

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 [74]:
```
from subprocess import call
call(['python', '-m', 'nbconvert', 'Analyze_ab_test_results_notebook.ipynb'])
```

Out[74]:
```
0
```