Analyze_ab_test_results_notebook

February 19, 2018

0.1 Analyze A/B Test Results

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

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 [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:

```
In [2]: #Load the dataset and take a look of the first few lines
        df = pd.read_csv('ab_data.csv')
        df.head()
```

```
Out[2]:    user_id                   timestamp      group landing_page  converted
        0   851104  2017-01-21 22:11:48.556739    control     old_page          0
        1   804228  2017-01-12 08:01:45.159739    control     old_page          0
        2   661590  2017-01-11 16:55:06.154213  treatment     new_page          0
        3   853541  2017-01-08 18:28:03.143765  treatment     new_page          0
        4   864975  2017-01-21 01:52:26.210827    control     old_page          1
```

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

```
In [3]: #look as the shape of the dataframe
        df.shape
```

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

c. The number of unique users in the dataset.

```
In [4]: #Check the unique user_id
        df.user_id.nunique()
```

```
Out[4]: 290584
```

d. The proportion of users converted.

```
In [5]: #Calculate the proportion of users converted
        df['converted'].mean()
```

```
Out[5]: 0.11965919355605512
```

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

```
In [6]: #Calculate the total rows with mispalced labels
        mis_1 = df.query('landing_page == "new_page" & group != "treatment"').count()['user_id
        mis_2 = df.query('landing_page == "old_page" & group != "control"').count()['user_id']
        mis_1 + mis_2
```

```
Out[6]: 3893
```

f. Do any of the rows have missing values?

```
In [7]: #Check if there's any row missing values
        df.isnull().values.any()
```

```
Out[7]: 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. 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 [8]: #Filter the df dataframe to get the new df2 without mismatched data
        df2= df[(df['landing_page'] == 'new_page') & (df['group'] == 'treatment') | (df['landi
```

```
In [9]: # Double Check all of the correct rows were removed - this should be 0
        df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].sl
```

```
Out[9]: 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 [10]: #Get the unique user_id in df2
         df2.user_id.nunique(), df2.shape
```

```
Out[10]: (290584, (290585, 5))
```

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

```
In [11]: #Get the duplicate
         df2[df2.duplicated('user_id')]
```

Out[11]:

	user_id	timestamp	group	landing_page	converted
2893	773192	2017-01-14 02:55:59.590927	treatment	new_page	0

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

```
In [12]: #Retrive the duplicates based on the user_id
         df2[df2['user_id'] == 773192]
```

Out[12]:

	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 [13]: #drop the duplicated row
         df2 = df2.drop_duplicates()
```

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 [14]: print("The probability of converted:", df.converted.mean())
```

```
The probability of converted: 0.11965919355605512
```

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

```
In [15]: print("The probability of converted in control group:", df2.query('group == "control"
```

```
The probability of converted in control group: 0.1203863045004612
```

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

```
In [16]: print("The probability of converted in treatment group:", df2.query('group == "treatme
```

```
The probability of converted in treatment group: 0.11880724790277405
```

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

```
In [17]: print("The probability of converted in control group:", len(df2.query('landing_page ==
```

```
The probability of converted in control group: 0.5000636646764286
```

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

According to the results, the old control page has a 0.2% higher conversion rate comparing to the new treatment page. However, the results is not practically significant due to such a small difference. The results are not favorable to new treatment page either as the new treatment page actually has a lower conversion rate than the old version. But the naive probability test is far from enough to examine the true statistical significance. Further tests are needed.

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.

$$H_0 : p_{new} - p_{old} \leq 0$$
$$H_1 : p_{new} - p_{old} > 0$$

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

```
In [18]: #Calculating the conversion rate of the new page
         p_new = df2.query('landing_page == "new_page"')['converted'].mean()
         print('The convert rate for the new page:', p_new)
```

```
The convert rate for the new page: 0.11880724790277405
```

```
In [19]: #Calculating the conversion rate of the old page
         p_old = df2.query('landing_page == "old_page"')['converted'].mean()
         print('The convert rate for the new page:', p_old)
```

```
The convert rate for the new page: 0.1203863045004612
```

```
In [20]: print('The difference between the two conversion rate:', p_new - p_old)
```

```
The difference between the two conversion rate: -0.0015790565976871451
```

```
In [21]: #Calculating the mean conversion rate
         p_mean = np.mean([p_new, p_old])
         print('The mean conversion rate of p_new and p_old:',p_mean)
```

```
The mean conversion rate of p_new and p_old: 0.119596776202
```

According to the assumption provided above, under the null hypothesis, the p_new and p_old both have the same value, which means:

$$p_{new} = p_{old}$$

Thus, the p_new and p_old should both equal to the mean conversion rate of the dataset ad_data.csv regardless of the page type, which is 0.1196. So that:

$$p_{new} = 0.1196$$

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

The p_old under the null hypothesis should be as same as p_new, which is 0.1196.

c. What is n_{new}?

```
In [22]: #Get the n_new and n_old
         n_new, n_old = df2.landing_page.value_counts()
         print('n_new is:',n_new)
```

```
n_new is: 145311
```

d. What is n_{old}?

```
In [23]: print('n_old is:', n_old)
```

```
n_old is: 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]: #Simulate the new page converted on the size of the new pages
         new_page_converted = np.random.choice([1, 0], size = n_new, p = [p_mean, (1-p_mean)],
```

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]: #Simulate the old page converted on the size of the new pages
         old_page_converted = np.random.choice([1, 0], size = n_old, p = [p_mean, (1-p_mean)],
```

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

```
In [26]: #Calculate the sample mean difference
         new_page_converted.mean() - old_page_converted.mean()
```

```
Out[26]: -0.00033342429828678299
```

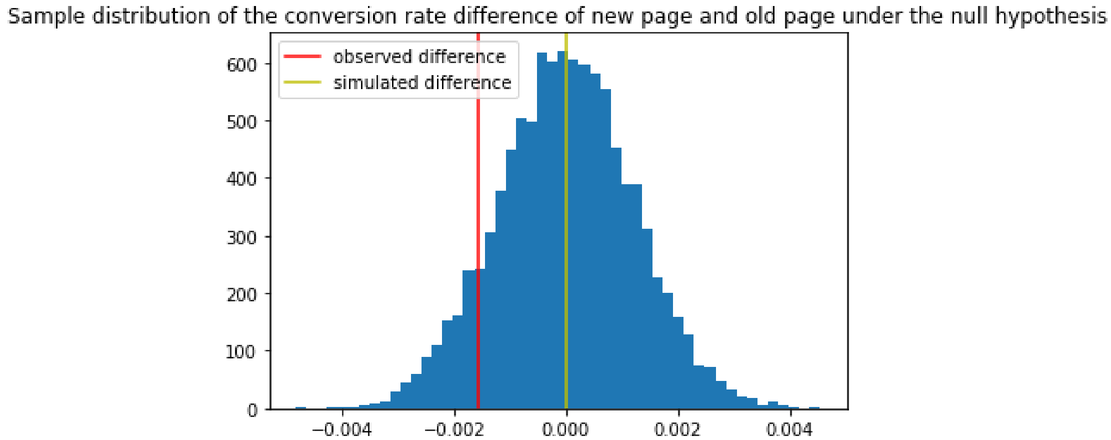
h. Simulate 10,000 p_{new} - p_{old} values using this same process similarly to the one you calculated in parts **a. through g.** above. Store all 10,000 values in a numpy array called **p_diffs**.

```
In [27]: #Generate a sample distribution based on the sample mean difference
         p_diffs = []
         for i in range(10000):
             new_page_converted = np.random.choice([1,0], size = n_new, p = [p_mean, (1-p_mean
             old_page_converted = np.random.choice([1,0], size = n_old, p = [p_mean, (1-p_mean
             p_diff = new_page_converted.mean() - old_page_converted.mean()
             p_diffs.append(p_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]: #Plot the histogram
         p_diffs = np.array(p_diffs)
         plt.hist(p_diffs, bins = 50)
         plt.title('Sample distribution of the conversion rate difference of new page and old p
         plt.axvline(x = p_new - p_old, color = 'r', label = 'observed difference')
         plt.axvline(x = np.mean(p_diffs), color = 'y', label = 'simulated difference')
         plt.legend()
```

Out[28]: <matplotlib.legend.Legend at 0x104b70518>



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

```
In [29]: #Get the proporation of the p_diffs greater than actual difference
         p_val = (p_diffs > (p_new-p_old)).mean()
         print('The proportion of p_diffs greater than the actual difference:',p_val)
```

```
The proportion of p_diffs greater than the actual difference: 0.9022
```

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?

The value result of the part j is called p value. It means that assuming the null hypothesis is true, the probability of getting the observed statistics or more extreme favours the alternative hypothesis. In this case, the p value is 0.91, which is high enough for us to consider the null hypothesis is true given if the type I error rate is 5%. This suggests that the new page does not do better than the old page, the implementation of the new landing page can get even worse conversion rate 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 [30]: #import the necessary library and get the convert_new, convert_old, n_old and n_new
         import statsmodels.api as sm
```

```
convert_old = df2.query('landing_page == "old_page" & converted == 1').count()['user_i
convert_new = df2.query('landing_page == "new_page" & converted == 1').count()['user_i
n_old = df2[df2['landing_page'] == 'old_page'].count()['user_id']
n_new = df2[df2['landing_page'] == 'new_page'].count()['user_id']
```

```
/anaconda3/lib/python3.6/site-packages/statsmodels/compat/pandas.py:56: FutureWarning: The panc
  from pandas.core import datetools
```

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 [57]: #Utilizing the z test to get the z score and p value
         z_score, p_value = sm.stats.proportions_ztest([convert_new, convert_old], [n_new, n_ol
```

```
In [58]: z_score, p_value
```

```
Out[58]: (-1.3116075339133115, 0.905173705140591 03)
```

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.**?

Z score measures the number of standard deviation of a data point differ from the mean. According to the above results, we can see that the difference between the mean from the null hypothesis and the observed mean is 1.31 standard deviations. The p value is 0.905, which is higher than 0.05, which indicates that we fail to reject the null hypothesis. The difference in the p value suggests the difference in the hypothesis in parts j. and k.. The null hypothesis for z-test is the conversion rate of the old page equals to the conversion rate of the new page, which indicates the alternative hypothesis is that the coversion rate of the old page is different from the conversion rate of the new page. As the results shown, the p value of the z test is 0.905, which means that we fail to reject the null hypothesis. The result agrees with the findings in part 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?

Due to the fact that the dependent variables are categorical variables, it is better for us to use logistic regression than other regression models.

Previous to implementing the regression model. There might be a valuable perspective to consider when analyzing A/B test results. Due to psychological reasons, the change from a old page to a new page may have different effect on different groups of users. There might be change aversion that prevent the old users to like the new page even the new page is indeed better than the old page, which will decrease the conversion rate for the old users in a certain period of time after the change being implemented. The novelty effect may affect the result as well, which means there are certain type of users who love the new changes regardless whether the new page brings positive changes or not. Fortunately, these psychological effects can be reduced or even eliminated after a certain period of time naturally. So it is important to add time variables to the regression model to see the effect of time.

First we want to see the data type of the time and convert it into a more accessible version for analysis.

```
In [33]: #Check to see the data type of the dataframe
         df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 290585 entries, 0 to 294477
Data columns (total 5 columns):
user_id         290585 non-null int64
timestamp       290585 non-null object
group           290585 non-null object
landing_page    290585 non-null object
converted       290585 non-null int64
dtypes: int64(2), object(3)
memory usage: 13.3+ MB
```

```
In [34]: #Convert the data type to datetime and then parse the date
         df2['date'] = pd.to_datetime(df2['timestamp'])
         df2['date'] = pd.DatetimeIndex(df2.date).normalize()
```

```
In [35]: #Get the sorted array of date
         np.sort(df2.date.unique())
```

```
Out[35]: array(['2017-01-02T00:00:00.000000000', '2017-01-03T00:00:00.000000000',
                '2017-01-04T00:00:00.000000000', '2017-01-05T00:00:00.000000000',
                '2017-01-06T00:00:00.000000000', '2017-01-07T00:00:00.000000000',
                '2017-01-08T00:00:00.000000000', '2017-01-09T00:00:00.000000000',
                '2017-01-10T00:00:00.000000000', '2017-01-11T00:00:00.000000000',
                '2017-01-12T00:00:00.000000000', '2017-01-13T00:00:00.000000000',
                '2017-01-14T00:00:00.000000000', '2017-01-15T00:00:00.000000000',
                '2017-01-16T00:00:00.000000000', '2017-01-17T00:00:00.000000000',
                '2017-01-18T00:00:00.000000000', '2017-01-19T00:00:00.000000000',
                '2017-01-20T00:00:00.000000000', '2017-01-21T00:00:00.000000000',
                '2017-01-22T00:00:00.000000000', '2017-01-23T00:00:00.000000000',
                '2017-01-24T00:00:00.000000000'], dtype='datetime64[ns]')
```

```
In [36]: #Get the sorted series of date to see the sample size for each date
         df2.date.value_counts().sort_index(axis = 0)
```

```
Out[36]: 2017-01-02     5712
         2017-01-03    13208
         2017-01-04    13119
         2017-01-05    12932
         2017-01-06    13353
```

```
2017-01-07    13213
2017-01-08    13387
2017-01-09    13243
2017-01-10    13350
2017-01-11    13361
2017-01-12    13159
2017-01-13    13060
2017-01-14    13148
2017-01-15    13263
2017-01-16    13136
2017-01-17    13155
2017-01-18    13085
2017-01-19    13130
2017-01-20    13213
2017-01-21    13309
2017-01-22    13265
2017-01-23    13349
2017-01-24     7435
Name: date, dtype: int64
```

In [37]:
```
#Check to see the data type
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 290585 entries, 0 to 294477
Data columns (total 6 columns):
user_id         290585 non-null int64
timestamp       290585 non-null object
group           290585 non-null object
landing_page    290585 non-null object
converted       290585 non-null int64
date            290585 non-null datetime64[ns]
dtypes: datetime64[ns](1), int64(2), object(3)
memory usage: 15.5+ MB
```

From the above array we can see that the dataset recorded 23 consecutive days of the behavior of users. So that there might be changes in the conversion rate over time. We can devide these dates into three different date groups.

In [38]:
```
#define a function to categorize data
def date_cat(date):
    date = np.datetime64(date)
    if date <= np.datetime64('2017-01-09'):
        return 'Begining'
    elif date <= np.datetime64('2017-01-17'):
        return 'Middle'
    elif date <= np.datetime64('2017-01-24'):
        return 'End'
```

```
In [39]: #Assign each date to a different group
         df2['dategroup'] = df2['date'].apply(date_cat)
```

```
In [40]: #Check to see if the function is working properly
         df2.head()
```

```
Out[40]:    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

                 date dategroup
         0 2017-01-21       End
         1 2017-01-12    Middle
         2 2017-01-11    Middle
         3 2017-01-08  Begining
         4 2017-01-21       End
```

```
In [41]: #Assigning dummie variables into the function for further analysis in regression mode
         df2[['Begining', 'End', 'Middle']] = pd.get_dummies(df2['dategroup'])
         df2.head()
```

```
Out[41]:    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

                 date dategroup  Begining  End  Middle
         0 2017-01-21       End         0    1       0
         1 2017-01-12    Middle         0    0       1
         2 2017-01-11    Middle         0    0       1
         3 2017-01-08  Begining         1    0       0
         4 2017-01-21       End         0    1       0
```

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**.

```
In [42]: #Add intercept column to the dataframe
         df2['intercept'] = 1
         group_dummies = pd.get_dummies(df2['group'])
         new_df2 = df2.join(group_dummies)
         new_df2 = new_df2.rename(columns = {'treatment' : 'ab_page'})
         new_df2.head()
```

Out[42]:
```
    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

         date dategroup  Begining  End  Middle  intercept  control  ab_page
0 2017-01-21       End         0    1       0          1        1        0
1 2017-01-12    Middle         0    0       1          1        1        0
2 2017-01-11    Middle         0    0       1          1        0        1
3 2017-01-08  Begining         1    0       0          1        0        1
4 2017-01-21       End         0    1       0          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 [43]:
```python
#Fit into the regression model
lm = sm.Logit(new_df2['converted'], new_df2[['intercept','ab_page']])
reg_lm = lm.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 [44]:
```python
reg_lm.summary()
```

Out[44]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              converted   No. Observations:               290585
Model:                          Logit   Df Residuals:                   290583
Method:                           MLE   Df Model:                            1
Date:                Sun, 18 Feb 2018   Pseudo R-squ.:               8.085e-06
Time:                        02:08:28   Log-Likelihood:            -1.0639e+05
converged:                       True   LL-Null:                   -1.0639e+05
                                        LLR p-value:                    0.1897
==============================================================================
                 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     -1.312      0.190      -0.037       0.007
==============================================================================
"""
```