Analyze_ab_test_results_notebook

February 4, 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

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

```
In [3]: df.shape
```

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

c. The number of unique users in the dataset.

```
In [4]: df['user_id'].nunique()
```

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

d. The proportion of users converted.

```
In [5]: df['converted'].mean()
```

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

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

```
In [6]: df.query('(group == "treatment" and landing_page != "new_page") or (group != "treatment"
```

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

f. Do any of the rows have missing values?

```
In [7]: df.isnull().values.any()
```

```
Out[7]: False
```

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**.

```
In [8]: df2 = df.drop(df.query('(group == "treatment" and landing_page != "new_page") or (group
```

```
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].sha
```

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

```
Out[10]: 290584
```

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

```
In [11]: df2[df2.duplicated(['user_id'],keep=False)]['user_id']
```

```
Out[11]: 1899    773192
         2893    773192
         Name: user_id, dtype: int64
```

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

```
In [12]: 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]: df2 = df2.drop(df2.query('(user_id == "773192") and (timestamp == "2017-01-09 05:37:58.
```

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?

```
In [14]: df2['converted'].mean()
```

```
Out[14]: 0.11959708724499628
```

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

```
In [15]: treatment_converted = df2.query('group =="control" and converted == "1"').count()
         group_treatment = df2.query('group =="control"').count()
         treatment_conversion_rate = treatment_converted/group_treatment
         treatment_conversion_rate
```

```
Out[15]: user_id         0.120386
         timestamp       0.120386
         group           0.120386
         landing_page    0.120386
         converted       0.120386
         dtype: float64
```

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

```
In [16]: treatment_converted = df2.query('group =="treatment" and converted == "1"').count()
         group_treatment = df2.query('group =="treatment"').count()
         treatment_conversion_rate = treatment_converted/group_treatment
         treatment_conversion_rate
```

```
Out[16]: user_id         0.118808
         timestamp       0.118808
         group           0.118808
         landing_page    0.118808
         converted       0.118808
         dtype: float64
```

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

```
In [17]: new_page_total= df2.query('landing_page =="new_page"').count()
         new_page_prob = new_page_total/df2.shape[0]
         new_page_prob
```

```
Out[17]: user_id         0.500062
         timestamp       0.500062
         group           0.500062
         landing_page    0.500062
         converted       0.500062
         dtype: float64
```

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.

The probability of converting for an individual who received the control page is 0.1204. The probability of converting an individual who received the treatment page is 0.1188. The conversion rates are so close that there is no evidence that one page leads to more conversions than the other.

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.

p_{new} <=p_{old}

p_{new} > p_{old}

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?

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

Out[21]: 0.11959708724499628
```

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

```
In [22]: p_old = df2['converted'].mean()
         p_old

Out[22]: 0.11959708724499628
```

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

```
In [23]: n_new = df2.query('landing_page == "new_page"').shape[0]
         n_new

Out[23]: 145310
```

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

```
In [25]: n_old = df2.query('landing_page == "old_page"').shape[0]
         n_old

Out[25]: 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**.

```
In [26]: new_page_converted = np.random.binomial(n_new, p_new)
```

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**.

```
In [27]: old_page_converted = np.random.binomial(n_old, p_old)
```

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

```
In [29]: obs_diff = new_page_converted/n_new - old_page_converted/n_old
         obs_diff
```

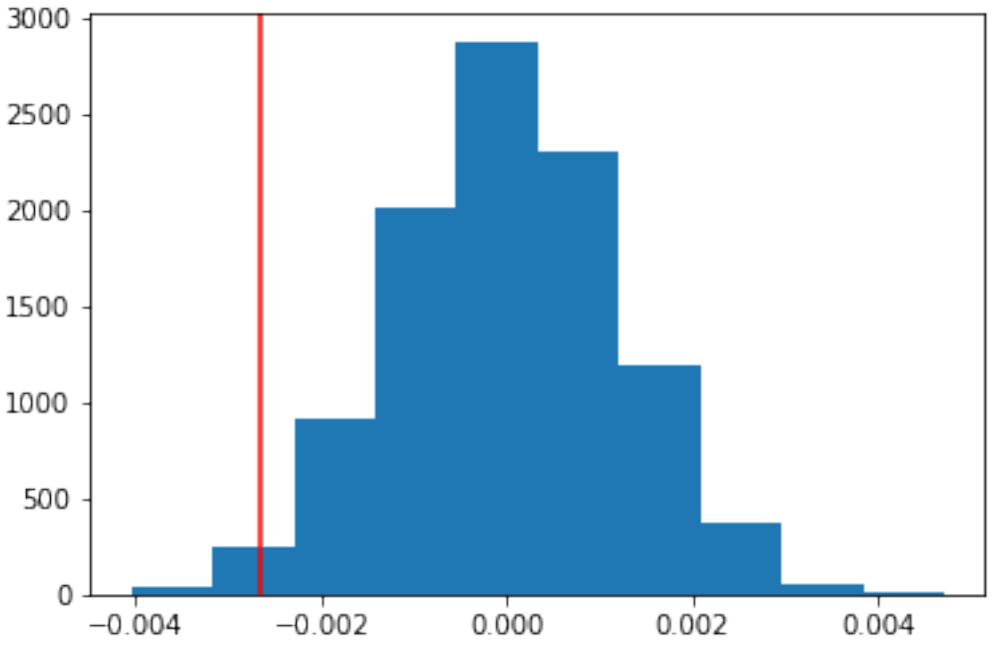
```
Out[29]: -0.0026585887696767563
```

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**.

```
In [30]: 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)
             p_diffs.append(new_page_converted/n_new - old_page_converted/n_old)
```

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 [31]: p_diffs = np.array(p_diffs)
         plt.hist(p_diffs);
         plt.axvline(x=obs_diff, color="red");
```



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

```
In [32]: actual_diff = df2.query('group == "treatment"')['converted'].mean() - df2.query('group
         actual_diff
```

```
Out[32]: -0.0015782389853555567
```

```
In [33]: p_val = (p_diffs > actual_diff).mean()
         p_val
```

```
Out[33]: 0.90510000000000002
```

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?

The part j computes the p value which is the probability of obtaining the observed statistic or a "more extreme" value if the null hypothesis is true. The computed p value is large, therefore, we fail to reject the null hypothesis.

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 [34]: import statsmodels.api as sm

         convert_old =  df2.query('group == "control"')['converted'].sum()
         convert_new =  df2.query('group == "treatment"')['converted'].sum()
         n_old =  df2.query('landing_page == "old_page"').shape[0]
         n_new =  df2.query('landing_page == "new_page"').shape[0]
```

```
/opt/conda/lib/python3.6/site-packages/statsmodels/compat/pandas.py:56: FutureWarning: The panda
  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 [35]: z_score, p_value = sm.stats.proportions_ztest([convert_new, convert_old], [n_new, n_old
         print(z_score, p_value)
```

```
-1.31092419842 0.905058312759
```

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-value is obtained from a one-tail test as we are checking if two proportions are statistically worse-than each other. Since the z-value of -1.31 does not exceed the critical value at 95% confidence - which is 1.96, we fail to reject the null hypothesis.

The p-value of 0.91 is larger than 0.05 and so we also fail to reject the null hypothesis using the p-value. In other words, the new page is statistically no different and nor better than the old page. We would expect their long-term performance to be similar to one another. The conclusion is the same as the findings in parts j. and k.

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**.

```
In [42]: import statsmodels.api as sm
         df2[['treatment', 'control']] = pd.get_dummies(df2['group'])
         df2['ab_page'] = df2['treatment']
         df2['intercept'] = 1
```

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.

```
In [45]: logit_mod = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])
         results = logit_mod.fit()

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

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

```
In [46]: results.summary()

Out[46]: <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, 01 Jan 2020   Pseudo R-squ.:               8.077e-06
Time:                        01:09:22   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     -2.0038      0.008   -247.146      0.000      -2.020      -1.988
ab_page        0.0150      0.011      1.311      0.190      -0.007       0.037
==============================================================================
"""
```

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**?

The p-value associated with ab_page is 0.190. The reason why it differs from the p-value in part2 is the difference in the null and alternative hypotheses. For the regression model, the null hypothesis is p_{new}=p_{old} and the alternative hypothesis is p_{new}!=p_{old}.

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?

The advantages of considering other factors to add into my regression model is that we would be able to find out other factors that may influence why a person converts or does not convert. An example is the demography of converts. Perhaps you find out that most of your converts are in a specific age bracket, or are a specific gender, or are from a specific country, you can then begin to use that information when you market or decide on designing a new page.

Although considering other factors that could help us determine the cause of converts beyond the new page versus the old page could provide further insight as to whether the new page really affects or doesn't affect the conversion rate, there are some disadvantages. For example, We would like x-variables to be related to the response, but not to be related to one another (i.e. multicollinearity). Below are further issues related to adding additional terms to the regression model:

```
Non-linearity of the response-predictor relationships
Correlation of error terms
Non-constant Variance and Normally Distributed Errors
Outliers/ High leverage points
Multicollinearity
```

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.

```
In [49]: countries = pd.read_csv('countries.csv')
         df_new = countries.set_index('user_id').join(df2.set_index('user_id'), how='inner')
         df_new.head()
```

Out[49]:

user_id	country	timestamp	group	landing_page	converted	treatment	control	ab_page	intercept
834778	UK	2017-01-14 23:08:43.304998	control	old_page	0	1	0	1	1
928468	US	2017-01-23 14:44:16.387854	treatment	new_page	0	0	1	0	1
822059	UK	2017-01-16 14:04:14.719771	treatment	new_page	1	0	1	0	1
711597	UK	2017-01-22 03:14:24.763511	control	old_page	0	1	0	1	1
710616	UK	2017-01-16 13:14:44.000513	treatment	new_page	0	0	1	0	1

```
In [50]: countries_dummies = pd.get_dummies(df_new['country'])
         df_countries = df_new.join(countries_dummies)
         df_countries = df_countries.drop(['country', 'US'], axis=1)
         df_countries.head()
```

Out[50]:

user_id	timestamp	group	landing_page	converted	treatment	control	ab_page	intercept	CA	UK
834778	2017-01-14 23:08:43.304998	control	old_page	0	1	0	1	1	0	1
928468	2017-01-23 14:44:16.387854	treatment	new_page	0	0	1	0	1	0	0
822059	2017-01-16 14:04:14.719771	treatment	new_page	1	0	1	0	1	0	1
711597	2017-01-22 03:14:24.763511	control	old_page	0	1	0	1	1	0	1
710616	2017-01-16 13:14:44.000513	treatment	new_page	0	0	1	0	1	0	1

```
In [52]: logit_mod2 = sm.Logit(df_countries['converted'], df_countries[['intercept', 'CA', 'UK']
         results = logit_mod2.fit()
         results.summary()
```

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

```
Out[52]: <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:                Wed, 01 Jan 2020   Pseudo R-squ.:               1.521e-05
         Time:                        01:33:36   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     -1.9967      0.007   -292.314      0.000      -2.010      -1.983
         CA            -0.0408      0.027     -1.518      0.129      -0.093       0.012
         UK             0.0099      0.013      0.746      0.456      -0.016       0.036
         ==============================================================================
         """
```

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 [60]: logit_mod3 = sm.Logit(df_countries['converted'], df_countries[['intercept', 'ab_page',
         results = logit_mod3.fit()
         results.summary()

Optimization terminated successfully.
         Current function value: 0.366113
         Iterations 6


Out[60]: <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, 01 Jan 2020   Pseudo R-squ.:               2.323e-05
         Time:                        01:38:34   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]
         ------------------------------------------------------------------------------
```

```
intercept     -2.0042      0.009   -224.560      0.000      -2.022      -1.987
ab_page        0.0149      0.011      1.307      0.191      -0.007       0.037
CA            -0.0408      0.027     -1.516      0.130      -0.093       0.012
UK             0.0099      0.013      0.743      0.457      -0.016       0.036
==============================================================================
"""
```

It doesn't appear that an interaction between country and page has any impact on the conversion rate because all of the p-values are still larger than 0.05.

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!

```
In [61]: from subprocess import call
         call(['python', '-m', 'nbconvert', 'Analyze_ab_test_results_notebook.ipynb'])
```

```
Out[61]: 0
```

```
In [ ]:
```