# Analyze_ab_test_results_notebook

May 20, 2020

## 0.1 Analyze A/B Test Results

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

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

## 0.2 Table of Contents

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

### Introduction

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

For this project, you will be working to understand the results of an A/B test run by an e-commerce website. Your goal is to work through this notebook to help the company understand if they should implement the new page, keep the old page, or perhaps run the experiment longer to make their decision.

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

#### Part I - Probability

To get started, let's import our libraries.

```
In [8]: 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 [9]: df = pd.read_csv('ab_data.csv')
        df.head()
```

```
Out[9]:    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 [10]: df.shape
```

```
Out[10]: (294478, 5)
```

c. The number of unique users in the dataset.

```
In [11]: df.user_id.nunique()
```

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

d. The proportion of users converted.

```
In [12]: display(df.converted.mean() * 100)
```

```
11.965919355605511
```

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

```
In [13]: ans = df.query('group == "control" and landing_page =="new_page" or group == "treatment
         ans
```

```
Out[13]: 3893
```

f. Do any of the rows have missing values?

```
In [14]: df.isnull().sum()
```

```
Out[14]: user_id         0
         timestamp       0
         group           0
         landing_page    0
         converted       0
         dtype: int64
```

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

a. Now use the answer to the quiz to create a new dataset that meets the specifications from the quiz. Store your new dataframe in **df2**.

```
In [15]: #We will slice the data according to the condition given and drop the indexes
         # and create a new df

         new_pg = df.query('group == "control" and landing_page == "new_page"')
         old_pg = df.query('group == "treatment" and landing_page == "old_page"')

         idx = new_pg.append(old_pg).index
         idx
```

```
Out[15]: Int64Index([    22,    240,    490,    846,    850,    988,   1198,   1354,
                      1474,   1877,
                     ...
                     293240, 293302, 293391, 293443, 293530, 293773, 293817, 293917,
                     294014, 294252],
                    dtype='int64', length=3893)
```

```
In [16]: df2 = df.drop(idx)

         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 [17]: # 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[17]: 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 [18]: df2['user_id'].nunique()
```

```
Out[18]: 290584
```

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

```
In [19]: df2[df2.user_id.duplicated()]
```

```
Out[19]:       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 [20]: df2[df2['user_id'] == 773192]
```

```
Out[20]:       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 [21]: df2.drop([2893], inplace=True)

         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 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 [22]: #We will take a mean of the 'converted' colummn to get a probability
         df2.converted.mean()
```

```
Out[22]: 0.11959708724499628
```

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

```
In [23]: ans4b = df2[df2['group'] == 'control']['converted'].mean()

         ans4b
```

```
Out[23]: 0.1203863045004612
```

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

```
In [24]: ans4c = df2[df2['group'] == 'treatment']['converted'].mean()

         ans4c

Out[24]: 0.11880806551510564
```

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

```
In [25]: ans4d = (df2['landing_page'] == 'new_page').mean()

         ans4d

Out[25]: 0.50006194422266881
```

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.

**Answer:**

- The probability results show us that the 'converted' churn for the control group is higher than that of the 'treatment' group
- The probability that an individual recieved the new page is about 0.500 which is ~ 0.5
- With this, there isn't sufficcient data to 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.

**Answer:** $H_0 : p_{new} <= p_{old}$ , $H_1 : p_{new} > p_{old}$

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

         p_new

Out[28]: 0.11959708724499628
```

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

```
In [29]: p_old = df2.converted.mean()

         p_old

Out[29]: 0.11959708724499628
```

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

```
In [31]: n_new = df2[df2.group == 'treatment'].user_id.count()

         n_new

Out[31]: 145310
```

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

```
In [32]: n_old = df2[df2['group'] == 'control'].user_id.count()

         n_old

Out[32]: 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 [76]: 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 [77]: 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 [78]: new_page_converted/n_new - old_page_converted/n_old

Out[78]: 0.00077563469070063285
```

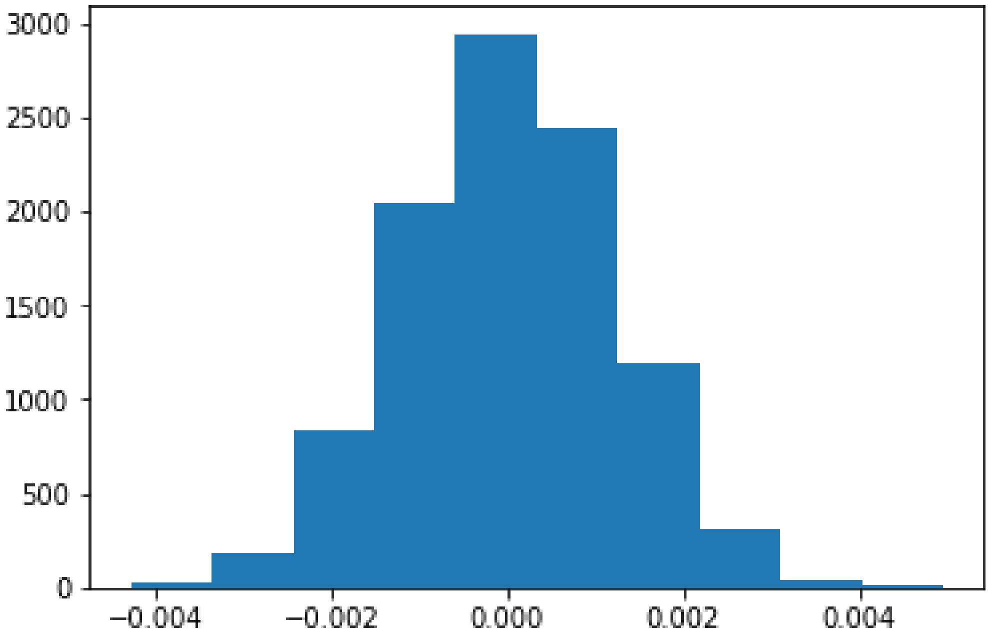
h. Create 10,000 $p_{new}$ - $p_{old}$ values using the same simulation process you used in parts (a) through (g) above. Store all 10,000 values in a NumPy array called **p_diffs**.

```
In [83]: p_diffs = []

         for _ in range(10000):
             sample = df2.sample(replace = True)
             new_page_sample = np.random.binomial(n_new,p_new)
             old_page_sample = np.random.binomial(n_old, p_old)
             diff = new_page_sample/n_new - old_page_sample/ 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 [84]: plt.hist(p_diffs);
```



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

```
In [91]: #lets calculate the actual difference
         act_diffs = df2[df2['group'] == 'treatment'].converted.mean() - df2[df2['group'] == 'co
         display(diffs)

         #Now let us calculate the p-value
         pval = (p_diffs > act_diffs).mean()
         pval
```

```
-0.0015782389853555567
```

```
Out[91]: 0.90649999999999997
```

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?

**Answer:**

- The value computed in part j in scientific studies is called the p-value.
- In scientific terms, the p-value or the probability value is the probability of obtaining the test results at least as extreme as the results actually observed during the test, assuming that the null hypothesis is correct.
- Looking at the p-value, we have evidence to fail to reject the null hypothesis testing, which means that the old page's coversion rate is higher than the new 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 [93]: 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[df2.group == 'control'].shape[0]
         n_new = df2[df2.group == 'treatment'].shape[0]

         display(n_old)
         display(n_new)
```

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

         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.**?

**Answer:**

- The z_score (i.e. 1.31) is less than the critical value of 1.64485.
- Hence, we fail to reject Null Hypothesis which suggests that Old page conversion is higher than the New page.
- Yes, I agree with the findings in the part j & 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?

**Answer:**

- Logistic Regression, since each row is either a conversion or no conversion (category) type we can clearly see its a logistic regression problem.

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 [95]: #. We will add an intercept column, as well as an ab_page column,
         # which is 1 when an individual receives the treatment and 0 if control.


         df2['intercept'] = 1
         df2[['ab_page2', 'ab_page']] = pd.get_dummies(df2['group'])
         df2[['control','treatment']] = pd.get_dummies(df2['group'])
         df2 = df2.drop('ab_page2', axis = 1)
         df2.head()
```

```
Out[95]:    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  ab_page  control  treatment
         0          1        0        1          0
         1          1        0        1          0
         2          1        1        0          1
         3          1        1        0          1
         4          1        0        1          0
```

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 [96]: import statsmodels.api as sm

         log_reg = sm.Logit(df2['converted'], df2[['intercept','ab_page']])
         results = log_reg.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 [97]: results.summary2()

Out[97]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                   Results: Logit
         =================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2020-05-20 04:46 AIC:              212780.3502
         No. Observations:   290584           BIC:              212801.5095
         Df Model:           1                Log-Likelihood:   -1.0639e+05
         Df Residuals:       290582           LL-Null:          -1.0639e+05
         Converged:          1.0000           Scale:            1.0000
         ------------------------------------------------------------------
                       Coef.   Std.Err.      z      P>|z|    [0.025   0.975]
         ------------------------------------------------------------------
         intercept    -1.9888    0.0081  -246.6690  0.0000  -2.0046  -1.9730
         ab_page      -0.0150    0.0114    -1.3109  0.1899  -0.0374   0.0074
         =================================================================

         """
```

e. What is the p-value associated with **ab_page**? Why does it differ from the value you found in **Part II**? **Hint**: What are the null and alternative hypotheses associated with your regression model, and how do they compare to the null and alternative hypotheses in **Part II**?

**Answer:**

- The p_value associated with ab_page is 0.1899 ~ 0.190

This one was a two-sided test, in Part-II it was one-sided test. We will test for the not equal type hypothesis.

1 : !=

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?

**Answer:**

- Other attributes like age or gender etc to the model can be unfruitful for our model.
- The effects of Simpson's paradox and similar phenomenon can come to play hence wont be so much useful.
- We lose some degree of freedom when we add a new predictor variable.
- There is a possibility of adding highly correlated predictors which can bring in Multicollinearity leading to unreliable and unstable results from our regression model.

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 [98]: 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[98]:         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  control  treatment
         user_id
         834778           0          1        0        1          0
         928468           0          1        1        0          1
         822059           1          1        1        0          1
         711597           0          1        0        1          0
         710616           0          1        1        0          1
```

```
In [99]: # Creating dummy variables

         df_new['intercept'] = 1
         df_new[['UK','US']] = pd.get_dummies(df_new['country'])[['UK','US']]
```

```
In [100]: df_new['US_ab_page'] = df_new['US'] * df_new['ab_page']
          df_new['UK_ab_page'] = df_new['UK'] * df_new['ab_page']
```

```
In [101]: df_new.head()
```

```
Out[101]:         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  control  treatment  UK  US  \
          user_id
          834778           0          1        0        1          0   1   0
          928468           0          1        1        0          1   0   1
          822059           1          1        1        0          1   1   0
          711597           0          1        0        1          0   1   0
          710616           0          1        1        0          1   1   0

                   US_ab_page  UK_ab_page
          user_id
          834778            0           0
          928468            1           0
          822059            0           1
          711597            0           0
          710616            0           1
```

```
In [102]: lm_mod = sm.Logit(df_new['converted'], df_new[['intercept', 'US', 'UK']])

          results = lm_mod.fit()

          results.summary2()
```

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

```
Out[102]: <class 'statsmodels.iolib.summary2.Summary'>
          """
                                    Results: Logit
          ==================================================================
          Model:              Logit            No. Iterations:   6.0000
          Dependent Variable: converted        Pseudo R-squared: 0.000
          Date:               2020-05-20 04:48 AIC:              212780.8333
          No. Observations:   290584           BIC:              212812.5723
          Df Model:           2                Log-Likelihood:   -1.0639e+05
          Df Residuals:       290581           LL-Null:          -1.0639e+05
          Converged:          1.0000           Scale:            1.0000
          -------------------------------------------------------------------
```

```
                 Coef.   Std.Err.     z      P>|z|    [0.025   0.975]
-------------------------------------------------------------------
intercept      -2.0375    0.0260  -78.3639  0.0000  -2.0885  -1.9866
US              0.0408    0.0269    1.5178  0.1291  -0.0119   0.0935
UK              0.0507    0.0284    1.7863  0.0740  -0.0049   0.1064
==================================================================

"""
```

**The Statistical analysis shows that the p-value for both the country attributes is larger than 0.05; hence indicating that there is no statistical evidence about it having a significant impact on the conversion.**

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 [104]: #  I have tried and included the ab_page column from part II exercise

          mod_2 = sm.Logit(df_new['converted'], df_new[['intercept', 'UK', 'US', 'ab_page', 'US_
```

```
In [105]: results= mod_2.fit()

          results.summary2()
```

```
Optimization terminated successfully.
         Current function value: 0.366109
         Iterations 6
```

```
Out[105]: <class 'statsmodels.iolib.summary2.Summary'>
          """
                                    Results: Logit
          ==================================================================
          Model:              Logit            No. Iterations:   6.0000
          Dependent Variable: converted        Pseudo R-squared: 0.000
          Date:               2020-05-20 04:50 AIC:              212782.6602
          No. Observations:   290584           BIC:              212846.1381
          Df Model:           5                Log-Likelihood:   -1.0639e+05
          Df Residuals:       290578           LL-Null:          -1.0639e+05
          Converged:          1.0000           Scale:            1.0000
          -------------------------------------------------------------------
                           Coef.   Std.Err.     z      P>|z|    [0.025   0.975]
          -------------------------------------------------------------------
          intercept      -2.0040    0.0364  -55.0077  0.0000  -2.0754  -1.9326
          UK              0.0118    0.0398    0.2957  0.7674  -0.0663   0.0899
          US              0.0175    0.0377    0.4652  0.6418  -0.0563   0.0914
```

```
ab_page       -0.0674    0.0520   -1.2967  0.1947  -0.1694   0.0345
US_ab_page     0.0469    0.0538    0.8718  0.3833  -0.0585   0.1523
UK_ab_page     0.0783    0.0568    1.3783  0.1681  -0.0330   0.1896
==================================================================

"""
```

**The output above shows that even after the including the column (which defines interaction between page and country) also there is no statistical evident to indicate that there is an impact on the conversion since p-values are all greater 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!

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