Analyze_ab_test_results_notebook

April 15, 2022

1 Analyze A/B Test Results

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)
```

Now, we read our 'ab_data' file

```
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.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 294478 entries, 0 to 294477
Data columns (total 5 columns):
user_id         294478 non-null int64
timestamp       294478 non-null object
group           294478 non-null object
landing_page    294478 non-null object
converted       294478 non-null int64
dtypes: int64(2), object(3)
memory usage: 11.2+ MB
```

So, we have 294478 rows

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

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

We have 294478 rows and 5 columns

c. The number of unique users in the dataset.

```
In [5]: len(pd.unique(df['user_id']))
```

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

We have 290584 unique users

d. The proportion of users converted.

```
In [6]: df.query('converted == 1').count()[0]/ df.shape[0]
```

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

e. The number of times when the "group" is `treatment` but "landing_page" is not a `new_page`.
This is obtained as follows:

```
In [7]: a =df[(df['group'] == 'treatment') & (df['landing_page'] != 'new_page')].count()[0]
        b = df[(df['group'] == 'control') & (df['landing_page'] != 'old_page')].count()[0]
        a + b
```

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

f. Do any of the rows have missing values?

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

```
Out[8]: user_id         294478
        timestamp       294478
        group           294478
        landing_page    294478
        converted       294478
        dtype: int64
```

There is no row having missing values

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**.

We will remove the inaccurate rows, and store the result in a new dataframe df2

```
In [9]: d1 =df[(df['group'] == 'treatment') & (df['landing_page'] == 'new_page')]
        d2 =df[(df['group'] == 'control') & (df['landing_page'] == 'old_page')]
        s = [d1,d2]
        d2.shape
```

```
Out[9]: (145274, 5)
```

Now, we concat s

```
In [10]: df2= pd.concat(s)
         df2.head()
```

```
Out[10]:    user_id                   timestamp      group landing_page  converted
         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
         6   679687  2017-01-19 03:26:46.940749  treatment     new_page          1
         8   817355  2017-01-04 17:58:08.979471  treatment     new_page          1
         9   839785  2017-01-15 18:11:06.610965  treatment     new_page          1
```

We will attempt to double check, if all of the incorrect rows were removed from df2

```
In [11]: df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].sh
```

```
Out[11]: 0
```

a. How many unique **user_id**s are in **df2**?

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

```
Out[12]: 290584
```

There are 290584 unique_ids in df2
b. There is one **user_id** repeated in **df2**. What is it?

```
In [13]: df2['user_id'].mode()
```

```
Out[13]: 0    773192
         dtype: int64
```

That is definitely user - 773192
c. Display the rows for the duplicate **user_id**?

```
In [14]: df2.loc[df2['user_id']== 773192]
```

```
Out[14]:       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
```

Those are the duplicate rows
d. Remove **one** of the rows with a duplicate **user_id**, from the **df2** dataframe.

```
In [15]: df2= df2.drop_duplicates(subset = 'user_id', keep = 'first')
```

To re-check again if the row with a duplicate user_id is deleted or not

```
In [16]: df2.loc[df2['user_id']== 773192]
```

```
Out[16]:       user_id                   timestamp      group landing_page  converted
         1899   773192  2017-01-09 05:37:58.781806  treatment     new_page          0
```

Great!!! It has been deleted
Now, let us check the shape of our dataset

```
In [17]:  df2.shape
```

```
Out[17]: (290584, 5)
```

.
a. What is the probability of an individual converting regardless of the page they receive?

```
In [18]: df2.head(3)
```

```
Out[18]:    user_id                   timestamp      group landing_page  converted
         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
         6   679687  2017-01-19 03:26:46.940749  treatment     new_page          1
```

```
In [19]: df2['converted'].sum()/290584
```

```
Out[19]: 0.11959708724499628
```

The probability an individual converts regardless of the page they recieve is 0.1196
b. Given that an individual was in the `control` group, what is the probability they converted?

```
In [20]: df2.groupby(['group', 'converted']).count()
```

```
Out[20]:                       user_id  timestamp  landing_page
         group     converted
         control   0           127785     127785        127785
                   1            17489      17489         17489
         treatment 0           128046     128046        128046
                   1            17264      17264         17264
```

```
In [21]: Pc=17489/145274
         Pc
```

```
Out[21]: 0.1203863045004612
```

Given an indidual is in the control group, the probability they converted is 0.1204
c. Given that an individual was in the `treatment` group, what is the probability they converted?

```
In [22]: Pt=17264/145310
         Pt
```

```
Out[22]: 0.11880806551510564
```

Given an indidual is in the treatment group, the probability they converted is 0.1188

Now, we calculate the actual difference (obs_diff) between the conversion rates for the two groups.

```
In [23]: obs_diff= df2[df2['group']=='treatment']['converted'].mean() - df2[df2['group']=='contr
         obs_diff

Out[23]: -0.0015782389853555567
```

So, our obs_diff is -0.00158

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

```
In [24]: df2.groupby(['landing_page']).count()

Out[24]:              user_id  timestamp   group  converted
         landing_page
         new_page       145310     145310  145310     145310
         old_page       145274     145274  145274     145274
```

From the above groupby method, the probability an individual recieved a new page is 0.5

```
In [25]: 145310/290584

Out[25]: 0.5000619442226688
```

e. Consider your results from parts (a) through (d) above, and explain below whether the new `treatment` group users lead to more conversions.

Looking at the convertion rate (Pc and Pt), it can be easily said that the new treatment group does not lead to more convertion rate as Pc>Pt. However, we also have to take into consideration the number of individuals in each group before making our conclusion. From the analysis above, we saw that the number of indivuals in the control group is 145274 and that of the treatment group is 145310. As the different is not that much, we can infer that the above conclusion is valid: the control group has more convertion rate.

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 be your null and alternative hypotheses (H_0 and H_1)?

You can state your hypothesis in terms of words or in terms of p_{old} and p_{new}, which are the "converted" probability (or rate) for the old and new pages respectively.

Anwser new <= pold

```
new > pold

or

new - pold <= 0

new - pold > 0
```

1.0.1 ToDo 2.2 - Null Hypothesis H_0 Testing

a. What is the **conversion rate** for p_{new} under the null hypothesis?

```
In [26]: Pnew = Pold =df2['converted'].sum()/290584
         Pnew

Out[26]: 0.11959708724499628
```

The conversion rate for Pnew is 0.1196

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

```
In [27]: Pnew = Pold =df2['converted'].sum()/290584
         Pold

Out[27]: 0.11959708724499628
```

The conversion rate for Pold is also 0.1196

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

```
In [28]: nnew = df2[df2['group']=='treatment'].count()[0]
         nnew

Out[28]: 145310
```

The number of individuals in the treatment group is 145310

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

```
In [29]: nold = df2[df2['group']=='control'].count()[0]
         nold

Out[29]: 145274
```

The number of individuals in the control group is 145274

e. Simulate Sample for the `treatment` Group Simulate n_{new} transactions with a conversion rate of p_{new} under the null hypothesis.

We simulate a Sample for the treatment Group

```
In [35]: new_page_converted = np.random.choice([0,1], size = nnew)
         new_page_converted

Out[35]: array([1, 1, 1, ..., 1, 0, 0])
```

f. Simulate Sample for the `control` Group Simulate n_{old} transactions with a conversion rate of p_{old} under the null hypothesis.

We simulate a Sample for the control Group

```
In [36]: old_page_converted = np.random.choice([0,1], size = nold)
         old_page_converted

Out[36]: array([0, 1, 1, ..., 0, 1, 0])
```

g. Find the difference in the "converted" probability $(p'_{new}$ - $p'_{old})$ for your simulated samples from the parts (e) and (f) above.

Let "pUnew" be the treatment group converted probability of the simulated sample above

```
In [37]: pUnew = new_page_converted.mean()
         pUnew

Out[37]: 0.49943568921615855
```

Let "pUold" be the treatment group converted probability of the simulated sample above

```
In [38]: pUold = old_page_converted.mean()
         pUold

Out[38]: 0.50041301265195426
```

Let dU be the difference in the two converted probabilities, then:

```
In [39]: dU = pUnew- pUold
         dU

Out[39]: -0.00097732343579570724
```

h. Sampling distribution Re-create `new_page_converted` and `old_page_converted` and find the $(p'_{new}$ - $p'_{old})$ value 10,000 times using the same simulation process you used in parts (a) through (g) above.

Store all $(p'_{new}$ - $p'_{old})$ values in a NumPy array called `p_diffs`.

We create the sampling distribution as follows:

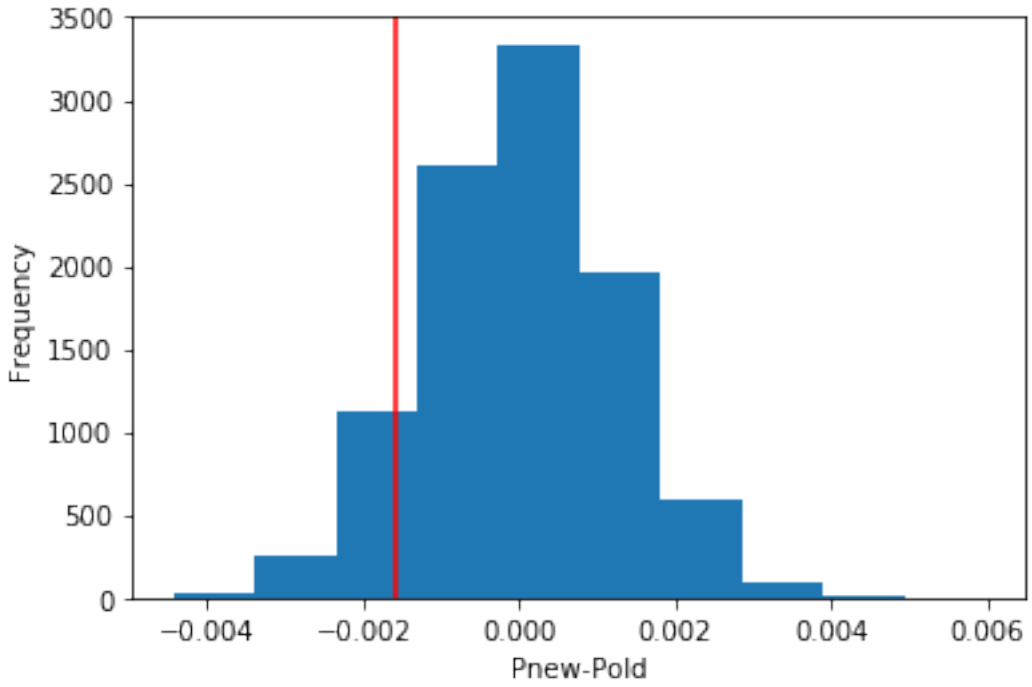
```
In [40]: p_diffs = []
         for _ in range (10000):
             new_page_converted = np.random.choice([0,1], size = nnew, replace = True, p = [Pnew
             old_page_converted = np.random.choice([0,1], size = nold, replace = True, p = [Pold
             pUnew = new_page_converted.mean()
             pUold = old_page_converted.mean()
             dU = pUnew- pUold
             p_diffs.append(dU)
```

i. Histogram 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.

We create our histogram as follows:

```
In [41]: plt.hist(p_diffs)
         plt.axvline(x=obs_diff, color = 'red')
         plt.ylabel('Frequency')
         plt.xlabel('Pnew-Pold')

Out[41]: Text(0.5,0,'Pnew-Pold')
```



As seen above, our histogram looks like what we expected.

j. What proportion of the **p_diffs** are greater than the actual difference observed in the `df2` data?

The required proportion is obtained as follows:

```
In [43]: (p_diffs > obs_diff).mean()

Out[43]: 0.90290000000000004
```

k. Please explain in words what you have just computed in part **j** above.
- What is this value called in scientific studies?
- What does this value signify in terms of whether or not there is a difference between the new and old pages?

Anwsers: 1. The P_value 2. Since our P_value(0.9029) is greater than our Type 1 error rate(0.05), we will fail to reject the null hypothesis. This implies that pnew <= Pold

l. Using Built-in Methods for Hypothesis Testing 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 walk-through of the ideas that are critical to correctly thinking about statistical significance.

```
In [44]: import statsmodels.api as sm

         # number of conversions with the old_page
         convert_old = df2[df2.group == 'control'].converted.sum()

         # number of conversions with the new_page
         convert_new =df2[df2.group == 'treatment'].converted.sum()
```

```
# number of individuals who were shown the old_page
n_old = df2[df2['landing_page'] == 'old_page'].count()[0]

# number of individuals who received new_page
n_new = df2[df2['landing_page'] == 'new_page'].count()[0]
```

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

```
In [45]: convert_old, convert_new, n_old, n_new
```

```
Out[45]: (17489, 17264, 145274, 145310)
```

m. Now use `sm.stats.proportions_ztest()` to compute your test statistic and p-value

```
In [46]: count_array = np.array([convert_new, convert_old])
         nobs_array = np.array([n_new,n_old])
```

```
In [47]: import statsmodels.api as sm
         z_score, p_value = sm.stats.proportions_ztest(count_array, nobs_array, alternative = 'l
         print(z_score, p_value)
```

```
-1.31092419842 0.905058312759
```

Therefore, our z_score is -1.311 and our P_value is 0.9051

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 p_value obtained here is similar to that obtained in section (2.2 j) above. Since the p_values are both greater than the Type 1 error rate, we will fail to reject the null hypothesis. So, pnew <= Pold

Part III - A regression approach

a. Since each row in the `df2` data is either a conversion or no conversion, what type of regression should you be performing in this case?

Answer: Multiple Linear Regression

b. The goal is to use **statsmodels** library to fit the regression model you specified in part **a.** above to see if there is a significant difference in conversion based on the page-type a customer receives. However, you first need to create the following two columns in the `df2` dataframe: 1. `intercept` - It should be `1` in the entire column. 2. `ab_page` - It's a dummy variable column, having a value `1` when an individual receives the **treatment**, otherwise `0`.

```
In [48]: df2.head(2)
```

```
Out[48]:    user_id                   timestamp      group landing_page  converted
         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
```

The first thing to do is to create our intercept as follows:

```
In [49]: df2['intercept'] = 1
```

Then, we create our dummy variable for the 'landing_page' columnn, using 'old_page' as baseline as follows:

```
In [50]: df2[['new_page', 'old_page']] = pd.get_dummies(df['landing_page'])
```

Done!!!

```
In [51]: df2.head(2)
```

```
Out[51]:    user_id                   timestamp      group landing_page  converted  \
         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

            intercept  new_page  old_page
         2          1         1         0
         3          1         1         0
```

c. Use **statsmodels** to instantiate your regression model on the two columns you created in part (b). above, then fit the model to predict whether or not an individual converts.

We fit in our regression model like this:

```
In [52]: import statsmodels.api as sm
         lm = sm.OLS(df2['converted'], df2[['intercept','new_page']])
         results= lm.fit()
```

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

```
In [53]: results.summary()
```

```
Out[53]: <class 'statsmodels.iolib.summary.Summary'>
         """
                                   OLS Regression Results
         ==============================================================================
         Dep. Variable:              converted   R-squared:                       0.000
         Model:                            OLS   Adj. R-squared:                  0.000
         Method:                 Least Squares   F-statistic:                     1.719
         Date:                Fri, 15 Apr 2022   Prob (F-statistic):              0.190
         Time:                        07:40:06   Log-Likelihood:                -85267.
         No. Observations:              290584   AIC:                         1.705e+05
         Df Residuals:                  290582   BIC:                         1.706e+05
         Df Model:                           1
         Covariance Type:            nonrobust
         ==============================================================================
                          coef    std err          t      P>|t|      [0.025      0.975]
         ------------------------------------------------------------------------------
         intercept      0.1204      0.001    141.407      0.000       0.119       0.122
```

```
new_page       -0.0016      0.001     -1.311      0.190      -0.004       0.001
==============================================================================
Omnibus:                   125553.456   Durbin-Watson:                   2.000
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           414313.355
Skew:                           2.345   Prob(JB):                         0.00
Kurtosis:                       6.497   Cond. No.                         2.62
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specif
"""
```

e. What is the p-value associated with **ab_page**? Why does it differ from the value you found in **Part II**?

Hints: - 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**? - You may comment on if these hypothesis (Part II vs. Part III) are one-sided or two-sided. - You may also compare the current p-value with the Type I error rate (0.05).

Answer The p_value associated with ab_page(new_page) is 0.190. If differs from that obtained in Part 11 because:Part II uses z-test statistics to calculate the p-value and Part III uses logistic regression which calculates the p-value using t-test statistics.

Te difference between the z-test and the t-test (and consequent p-value calculation) is that the z-test was conducted with one-tailed test, while the t-test for logistic regression was a two tailed test.

The p_value of 0.190 is also greater than the Type 1 error rate(0.05), therefore, this suggests that the variable 'new_page' is not statistially significant in relating to the response variable(i.e convertibility)

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: In a real life situation, the factors that determines whether or not an indiviual converts are much more than those considered in the analysis above. Thus, a given analysis might not be totally accurate(biased) if those underlying factors are not considered.

However, if too many factors are being considered for a given analysis, it could lead to a problem of multicollinearity.

g. Adding countries Now along with testing if the conversion rate changes for different pages, also add an effect based on which country a user lives in.

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

Let us read our 'countries' data

```
In [54]: dfc = pd.read_csv('countries.csv')
         dfc.head(5)
```

Out[54]:

	user_id	country
0	834778	UK
1	928468	US
2	822059	UK
3	711597	UK
4	710616	UK

Now, we will attempt to merge this 'countries' dataset with our dataset df2 in df3 like this:

```
In [55]: df3= pd.merge(df2, dfc, on ='user_id')
         df3.head()
```

```
Out[55]:    user_id                   timestamp      group landing_page  converted  \
         0   661590  2017-01-11 16:55:06.154213  treatment     new_page          0
         1   853541  2017-01-08 18:28:03.143765  treatment     new_page          0
         2   679687  2017-01-19 03:26:46.940749  treatment     new_page          1
         3   817355  2017-01-04 17:58:08.979471  treatment     new_page          1
         4   839785  2017-01-15 18:11:06.610965  treatment     new_page          1

            intercept  new_page  old_page country
         0          1         1         0      US
         1          1         1         0      US
         2          1         1         0      CA
         3          1         1         0      UK
         4          1         1         0      CA
```

Now we will create dummy variables for the country column, using 'CA' as baseline

```
In [56]: df3[['US', 'UK', 'CA']] = pd.get_dummies(df3['country'])
```

h. Fit your model and obtain the results 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 are there significant effects on conversion.

First of all, let us create an interaction between the page and country variable as follows:

```
In [57]: df3['new_page_UK'] = df3['new_page'] * df3['UK']
         df3['new_page_US'] = df3['new_page'] * df3['US']
```

We fit your model, and summarize the results

```
In [58]: df3['intercept'] = 1
         lm = sm.Logit(df3['converted'], df3[['intercept', 'new_page', 'UK' , 'US', 'new_page_UK
         results= lm.fit()
         results.summary2()
```

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


Out[58]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                  Results: Logit
         ================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
```

```
Date:                2022-04-15 07:41 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    -1.9865    0.0096  -206.3440  0.0000  -2.0053  -1.9676
new_page     -0.0206    0.0137    -1.5052  0.1323  -0.0473   0.0062
UK           -0.0057    0.0188    -0.3057  0.7598  -0.0426   0.0311
US           -0.0175    0.0377    -0.4652  0.6418  -0.0914   0.0563
new_page_UK   0.0314    0.0266     1.1807  0.2377  -0.0207   0.0835
new_page_US  -0.0469    0.0538    -0.8718  0.3833  -0.1523   0.0585
==================================================================

"""
```

Anwser: Considering the P_values with a 0.05 Type 1 error rate: As the P-values for new_page(0.1323), UK(0.7598), US(0.6418), new_page_UK(0.2377) and new_page_US(0.3833) are all greater than 0.05, it means that none of this variables is statistically significant for predicting the conversion rate of individuals. We will fail to reject the null hypothesis.

In practical sense, it means that neither the location of an individual, nor the individual's landing page can be used to predict whether or not the individual will convert or not.

Final Check!

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 notebook to make sure that it satisfies all the specifications mentioned in the rubric. You should also probably remove all of the "Hints" and "Tips" like this one so that the presentation is as polished as possible.

Submission You may either submit your notebook through the "SUBMIT PROJECT" button at the bottom of this workspace, or you may work from your local machine and submit on the last page of this project lesson.

1. 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).

2. 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.

3. 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 [60]: from subprocess import call
         call(['python', '-m', 'nbconvert', 'Analyze_ab_test_results_notebook.ipynb'])

Out[60]: 0

In [ ]:
```