Analyze_ab_test_results_notebook

December 14, 2018

0.1 Analyze A/B Test Results

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

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

0.2 Table of Contents

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

Introduction

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

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

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

Part I - Probability

To get started, let's import our libraries.

```
In [424]: import pandas as pd
          import numpy as np
          import random
          import matplotlib.pyplot as plt
          %matplotlib inline
          import statsmodels.api as sm
          from scipy.stats import norm
          #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 [425]: df = pd.read_csv('ab_data.csv')
          df.head()
```

Out[425]:

	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 [426]: df.shape
```

```
Out[426]: (294478, 5)
```

c. The number of unique users in the dataset.

```
In [427]: df.user_id.nunique()
```

```
Out[427]: 290584
```

d. The proportion of users converted.

```
In [428]: df['converted'].mean()
```

```
Out[428]: 0.11965919355605512
```

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

```
In [429]: new_page = df.query('landing_page=="new_page"')
          new_lineup = new_page.query('group!="treatment"')
          treatment = df.query('group=="treatment"')
          treatment_lineup = treatment.query('landing_page!="new_page"')
          new_lineup.shape[0]+treatment_lineup.shape[0]
```

```
Out[429]: 3893
```

f. Do any of the rows have missing values?

```
In [430]: df.isnull().any()
```

```
Out[430]: user_id         False
          timestamp       False
          group           False
          landing_page    False
          converted       False
          dtype: bool
```

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 [431]: df2=df[((df.landing_page=='new_page') & (df.group=='treatment')) |
                 ((df.landing_page=='old_page') & (df.group=='control')) ]
```

```
In [432]: # Double Check all of the correct rows were removed - this should be 0
          df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].s
```

```
Out[432]: 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 [433]: df2.user_id.unique().shape[0]
```

```
Out[433]: 290584
```

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

```
In [434]: df2[df2.user_id.duplicated()]
```

```
Out[434]:       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 [435]: df2[df2['user_id'].duplicated(keep=False)]
```

```
Out[435]:       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 [436]: df2 = df2.drop_duplicates(subset='user_id')
          df2.shape
```

```
Out[436]: (290584, 5)
```

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 [437]: df2.converted.mean()
```

```
Out[437]: 0.11959708724499628
```

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

```
In [438]: p_control_converted = df2.query('group == "control"')['converted'].mean()
          p_control_converted

Out[438]: 0.1203863045004612
```

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

```
In [439]: p_treatment_converted = df2.query('group == "treatment"')['converted'].mean()
          p_treatment_converted

Out[439]: 0.11880806551510564
```

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

```
In [440]: len(df2.query('landing_page == "new_page"')) / len(df2['landing_page'])

Out[440]: 0.5000619442226688
```

e. Use the results in the previous two portions of this question to suggest if you think there is evidence that one page leads to more conversions? Write your response below.

There is no evidence that one page leads to more conversions, the conversion rate for both groups is the same, and the probability for individual receives is 0.5 which means both have the same chance

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.

****$Nulls: p_{new} - p_{old} > 0$****

****$Alternative: p_{new} - p_{old} \leq 0$****

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

Out[441]: 0.11959708724499628
```

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

```
In [442]: p_old = df2['converted'].mean()
          p_old

Out[442]: 0.11959708724499628
```

c. What is n_{new}?

```
In [443]: n_new = df2.query('landing_page == "new_page"')['landing_page'].count()
          n_new

Out[443]: 145310
```

d. What is n_{old}?

```
In [444]: n_old = df2.query('landing_page == "old_page"')['landing_page'].count()
          n_old

Out[444]: 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 [445]: newPage_converted = np.random.choice([1, 0], size=n_new,
                                                 p=[p_new, (1-p_new)])
          newPage_converted.mean()

Out[445]: 0.11958571330259446
```

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 [446]: oldPage_converted = np.random.choice([1,0], size=n_old,
                                                 p=[p_old, (1-p_old)])
          oldPage_converted.mean()

Out[446]: 0.11928493742858323
```

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

```
In [447]: p_new - p_old

Out[447]: 0.0
```

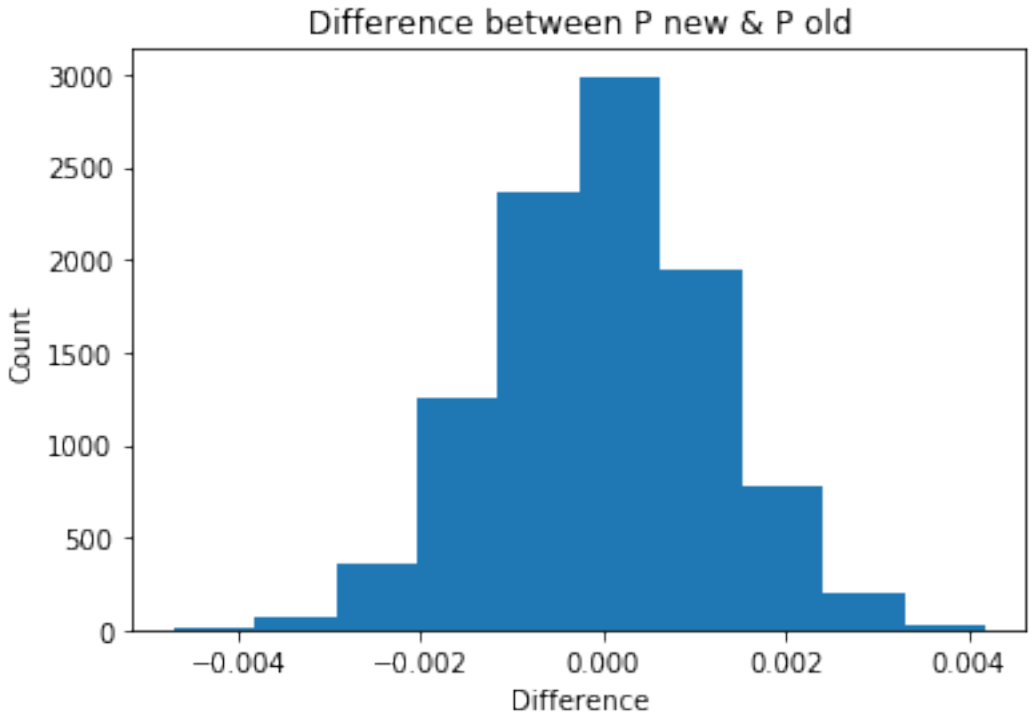
h. Simulate 10,000 p_{new} - p_{old} values using this same process similarly to the one you calculated in parts **a. through g.** above. Store all 10,000 values in **p_diffs**.

```
In [448]: p_diffs = []
          for i in range(10000):
              newPage_conv = np.random.choice([0, 1], n_new, p=[(1 - p_new), p_new])
              oldPage_conv = np.random.choice([0, 1], n_old, p=[(1 - p_old), p_old])
              p_diffs.append(newPage_conv.mean() - oldPage_conv.mean())

          p_diffs = np.array(p_diffs)
```

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 [449]: plt.hist(p_diffs);
          plt.title("Difference between P new & P old");
          plt.xlabel("Difference");
          plt.ylabel("Count");
          plt.show()
```



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

```
In [450]: (p_diffs > (p_treatment_converted - p_control_converted)).mean()
```

```
Out[450]: 0.90569999999999995
```

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 p-value was coputed in part J, 90% of the values are greater than the mean, which means 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 [451]: convert_old = df2.query('landing_page == "old_page"')['converted'].sum()
          convert_new = df2.query('landing_page == "new_page"')['converted'].sum()
          n_old = len(df2.query('landing_page == "old_page"'))
          n_new = len(df2.query('landing_page == "new_page"'))
```

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 [452]: z_score, p_value = sm.stats.proportions_ztest([convert_old, convert_new], [n_old, n_ne
          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 p-value = 0.90 which is greater than 0.5 (critical p-value), Due to this we reject the alternative hypothesis

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 binary dependent variable we'll be using the Logestic Regression

b. The goal is to use **statsmodels** to fit the regression model you specified in part **a.** to see if there is a significant difference in conversion based on which page a customer receives. However, you first need to create a colun for the intercept, and create a dummy variable column for which page each user received. Add an **intercept** column, as well as an **ab_page** column, which is 1 when an individual receives the **treatment** and 0 if **control**.

```
In [453]: df2['intercept']= 1
          df2[['control', 'treatment']] = pd.get_dummies(df['group'])
          df2 = df2.drop(['control'], axis=1)
          df.head()
```

Out[453]:

	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

```
In [454]: df2 = df2.rename(columns={'treatment': 'ab_page'})
          df2.head()
```

Out[454]:

	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
0	1	0
1	1	0
2	1	1
3	1	1
4	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 [455]: logist_reg = sm.Logit(df2['converted'],df2[['intercept','ab_page']])
```

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

```
In [456]: mod_summ = logist_reg.fit()
          mod_summ.summary()

Optimization terminated successfully.
         Current function value: 0.366118
         Iterations 6


Out[456]: <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:                Fri, 14 Dec 2018   Pseudo R-squ.:               8.077e-06
Time:                        14:44:07   Log-Likelihood:            -1.0639e+05
converged:                       True   LL-Null:                   -1.0639e+05
                                        LLR p-value:                    0.1899
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -1.9888      0.008   -246.669      0.000      -2.005      -1.973
ab_page       -0.0150      0.011     -1.311      0.190      -0.037       0.007
==============================================================================
"""
```

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

****$Nulls : p_{new} - p_{old} = 0$****
****$Alternative : p_{new} - p_{old} != 0$****
****Previously we were testing each condition sepertly, However, in this model we are testing both conditions****

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?

For this Regression Model we could add categorical variables such as : - Gender - Education - Geogarphical location - Mirtal status ..etc
However, we should carefully pick variables, because adding extra factors to the model would decrease the result accuracy, due to the variables correlation with each other.

g. Now along with testing if the conversion rate changes for different pages, also add an effect based on which country a user lives. You will need to read in the **countries.csv** dataset and merge together your datasets on the approporiate rows. Here are the docs for joining tables.

Does it appear that country had an impact on conversion? Don't forget to create dummy variables for these country columns - **Hint: You will need two columns for the three dummy varaibles.** Provide the statistical output as well as a written response to answer this question.

```
In [457]: df_countries = pd.read_csv('countries.csv')
          df_countries = df_countries.set_index('user_id')
          df_countries.head()

Out[457]:          country
          user_id
```

```
834778       UK
928468       US
822059       UK
711597       UK
710616       UK
```

In [458]:
```
df2 = df2.set_index('user_id')
df2.head()
```

Out[458]:
```
                         timestamp      group landing_page  converted  \
user_id
851104   2017-01-21 22:11:48.556739    control     old_page          0
804228   2017-01-12 08:01:45.159739    control     old_page          0
661590   2017-01-11 16:55:06.154213  treatment     new_page          0
853541   2017-01-08 18:28:03.143765  treatment     new_page          0
864975   2017-01-21 01:52:26.210827    control     old_page          1

         intercept  ab_page
user_id
851104           1        0
804228           1        0
661590           1        1
853541           1        1
864975           1        0
```

In [459]:
```
df_new = pd.merge(df2, df_countries, how='inner', left_index=True, right_index=True)
df_new.head()
```

Out[459]:
```
                         timestamp      group landing_page  converted  \
user_id
851104   2017-01-21 22:11:48.556739    control     old_page          0
804228   2017-01-12 08:01:45.159739    control     old_page          0
661590   2017-01-11 16:55:06.154213  treatment     new_page          0
853541   2017-01-08 18:28:03.143765  treatment     new_page          0
864975   2017-01-21 01:52:26.210827    control     old_page          1

         intercept  ab_page country
user_id
851104           1        0      US
804228           1        0      US
661590           1        1      US
853541           1        1      US
864975           1        0      US
```

In [460]:
```
df_new.country.unique()
```

Out[460]:
```
array(['US', 'CA', 'UK'], dtype=object)
```

In [461]:
```
country_dumm = pd.get_dummies(df_new['country'])
df_new = df_new.join(country_dumm)
df_new.head()
```

Out[461]:

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

user_id	intercept	ab_page	country	CA	UK	US
851104	1	0	US	0	0	1
804228	1	0	US	0	0	1
661590	1	1	US	0	0	1
853541	1	1	US	0	0	1
864975	1	0	US	0	0	1

In [462]:
```
logist_reg2 = sm.Logit(df_new['converted'], df_new[['intercept', 'CA', 'UK']])
mod_summ2 = logist_reg2.fit()
mod_summ2.summary()
```

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

Out[462]:

Logit Regression Results

Dep. Variable:	converted	No. Observations:	290584
Model:	Logit	Df Residuals:	290581
Method:	MLE	Df Model:	2
Date:	Fri, 14 Dec 2018	Pseudo R-squ.:	1.521e-05
Time:	14:44:08	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

In [463]:
```
1 /np.exp(-0.0408)
```

Out[463]: 1.0416437559600236

```
In [464]: np.exp(0.0099)

Out[464]: 1.0099491671175422
```

The p-value for CA =0.129, and for UK = 0.456
which aren't significant but > 0.5

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 [465]: df_new['UK_page'] = df_new['ab_page'] * df_new['UK']
          df_new['CA_page'] = df_new['ab_page'] * df_new['CA']
          df_new.head()

Out[465]:                          timestamp      group landing_page  converted  \
          user_id
          851104   2017-01-21 22:11:48.556739    control     old_page          0
          804228   2017-01-12 08:01:45.159739    control     old_page          0
          661590   2017-01-11 16:55:06.154213  treatment     new_page          0
          853541   2017-01-08 18:28:03.143765  treatment     new_page          0
          864975   2017-01-21 01:52:26.210827    control     old_page          1

                   intercept  ab_page country  CA  UK  US  UK_page  CA_page
          user_id
          851104           1        0      US   0   0   1        0        0
          804228           1        0      US   0   0   1        0        0
          661590           1        1      US   0   0   1        0        0
          853541           1        1      US   0   0   1        0        0
          864975           1        0      US   0   0   1        0        0

In [466]: logist_reg3 = sm.Logit(df_new['converted'], df_new[['intercept', 'UK', 'CA', 'ab_page'
          mod_summ3 = logist_reg3.fit()
          mod_summ3.summary()

Optimization terminated successfully.
         Current function value: 0.366109
         Iterations 6


Out[466]: <class 'statsmodels.iolib.summary.Summary'>
          """
                                    Logit Regression Results
          ==============================================================================
          Dep. Variable:              converted   No. Observations:               290584
          Model:                          Logit   Df Residuals:                   290578
          Method:                           MLE   Df Model:                            5
```

```
Date:                Fri, 14 Dec 2018   Pseudo R-squ.:               3.482e-05
Time:                        14:44:09   Log-Likelihood:           -1.0639e+05
converged:                       True   LL-Null:                  -1.0639e+05
                                        LLR p-value:                    0.1920
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -1.9865      0.010   -206.344      0.000      -2.005      -1.968
UK            -0.0057      0.019     -0.306      0.760      -0.043       0.031
CA            -0.0175      0.038     -0.465      0.642      -0.091       0.056
ab_page       -0.0206      0.014     -1.505      0.132      -0.047       0.006
UK_page        0.0314      0.027      1.181      0.238      -0.021       0.084
CA_page       -0.0469      0.054     -0.872      0.383      -0.152       0.059
==============================================================================
"""
```

```
In [467]: np.exp(mod_summ3.params)
```

```
Out[467]: intercept    0.137178
          UK           0.994272
          CA           0.982625
          ab_page      0.979646
          UK_page      1.031896
          CA_page      0.954198
          dtype: float64
```

None of the variables above has a p-value < 0.5, that means we have no statistical evidence to prove the relationship between the variables and landing page

Finishing Up

Congratulations! You have reached the end of the A/B Test Results project! This is the final project in Term 1. You should be very proud of all you have accomplished!

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

```
Out[469]: 0
```

```
In [ ]:
```