Analyze_ab_test_results_notebook

December 30, 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

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 [97]: 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 [100]: df = pd.read_csv('ab_data.csv')
          df.head()
```

Out[100]:

	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 [101]: df.shape
```

```
Out[101]: (294478, 5)
```

c. The number of unique users in the dataset.

```
In [102]: len(pd.value_counts(df["user_id"]))
```

```
Out[102]: 290584
```

d. The proportion of users converted.

```
In [103]: np.mean(df["converted"] / df["user_id"])
```

```
Out[103]: 1.5386640063012756e-07
```

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

```
In [104]: df.query('group=="treatment" and landing_page != "new_page" or group=="control" and la
```

```
Out[104]: user_id         3893
          timestamp       3893
          group           3893
          landing_page    3893
          converted       3893
          dtype: int64
```

f. Do any of the rows have missing values?

```
In [105]: df.isnull().sum()
```

```
Out[105]: user_id         0
          timestamp       0
          group           0
          landing_page    0
          converted       0
          dtype: int64
```

```
In [106]: #As we can see, there is no missing value
```

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 [107]: df2 = df[df[['group', 'landing_page']].apply(lambda x: x[0] == 'control' and x[1] == '
```

```
In [108]: df2.head()
```

```
Out[108]:    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 [109]: # 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[109]: 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 [110]: #df2.user_id.unique()
          len(pd.value_counts(df2["user_id"]))
```

```
Out[110]: 290584
```

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

```
In [111]: df2[df2['user_id'].duplicated()].count()
```

```
Out[111]: user_id         1
          timestamp       1
          group           1
          landing_page    1
          converted       1
          dtype: int64
```

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

```
In [112]: df2[df2.duplicated('user_id', keep=False)==True]
```

```
Out[112]:       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 [113]: df2.drop(df.index[2893], axis=0, inplace=True)
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
  """Entry point for launching an IPython kernel.
```

```
In [114]: #rechick for duplicates
          df2[df2['user_id'].duplicated()].count()
```

```
Out[114]: user_id         0
          timestamp       0
          group           0
          landing_page    0
          converted       0
          dtype: int64
```

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 [115]: df2['converted'].mean()
```

```
Out[115]: 0.11959708724499628
```

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

```
In [116]: df2.query('group== "control"')['converted'].mean()
```

```
Out[116]: 0.1203863045004612
```

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

```
In [117]: df2.query('group== "treatment"')['converted'].mean()
```

```
Out[117]: 0.11880806551510564
```

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

```
In [118]: df2.query('landing_page== "new_page"').count()[0]/df2.shape[0]
```

```
Out[118]: 0.50006194422266881
```

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 a very small diffrance(about 0.2 %) between control group and treatment group. Therefore, we can sa there is no evidance that a 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.

- 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.

```
In [119]: p_new = df2['converted'].mean()
          p_new

Out[119]: 0.11959708724499628
```

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

```
In [120]: p_old = df2['converted'].mean()
          p_old

Out[120]: 0.11959708724499628
```

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

```
In [121]: p_old = df2['converted'].mean()
          p_old

Out[121]: 0.11959708724499628
```

c. What is n_{new}?

```
In [122]: n_new= df2[(df2.landing_page == 'new_page')].shape[0]
          n_new

Out[122]: 145310
```

d. What is n_{old}?

```
In [123]: n_old = df2[(df2.landing_page == 'old_page')].shape[0]
          n_old

Out[123]: 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 [124]: new_page_converted = np.random.choice([1, 0], size=n_new, p=[p_new, (1-p_new)])
          new_page_converted

Out[124]: array([0, 0, 0, ..., 0, 0, 0])
```

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 [125]: old_page_converted = np.random.choice([1, 0], size=n_old, p=[p_old, (1-p_old)])
          old_page_converted

Out[125]: array([0, 0, 1, ..., 0, 0, 0])
```

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

```
In [126]: new_page_converted.mean()-old_page_converted.mean()

Out[126]: -0.00014659140017879435
```

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**.

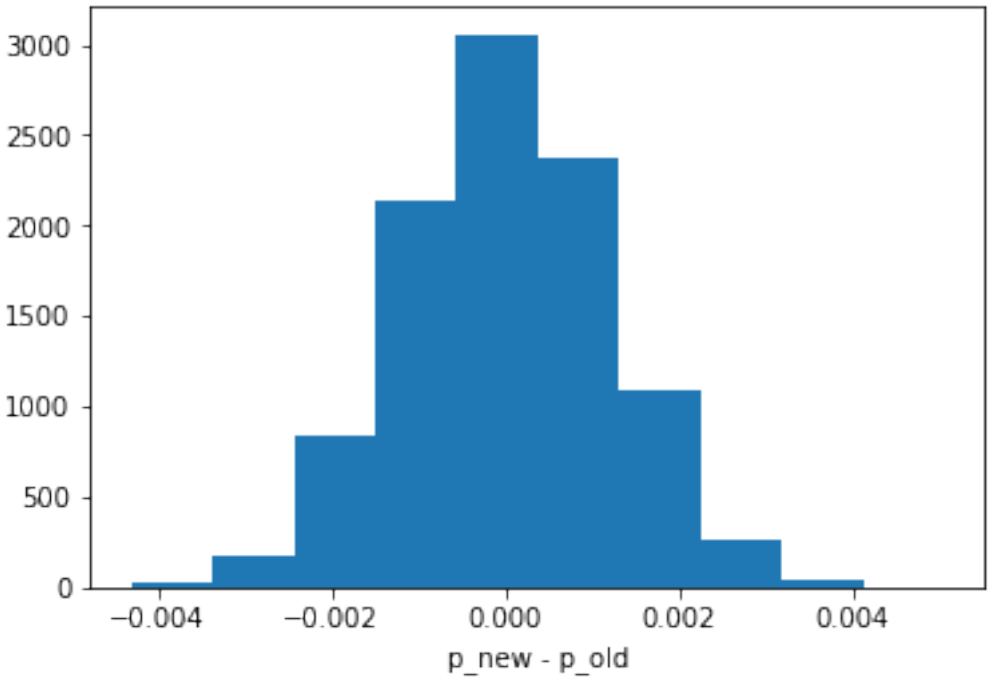
```
In [127]: p_diffs = []


          for i in range(10000):
              new_page_converted = np.random.choice([1, 0], size=n_new, p=[p_new, (1-p_new)]).me
              old_page_converted = np.random.choice([1, 0], size=n_old, p=[p_old, (1-p_old)]).me
              diff = new_page_converted - old_page_converted
              p_diffs.append(new_page_converted.mean() - old_page_converted.mean())
```

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 [128]: plt.hist(p_diffs)
          plt.xlabel('p_new - p_old')
```

```
Out[128]: Text(0.5,0,'p_new - p_old')
```



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

```
In [96]: obs_diffs = df[df['group'] == 'treatment']['converted'].mean() - df[df['group'] == 'con
         (p_diffs > obs_diffs).mean()
```

```
Out[96]: 0.89000000000000001
```

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?

P-value which is the probability of finding extreme results when the null hypothesis true is calculated. Since P-value is larger than 0.05 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 [129]: import statsmodels.api as sm

          convert_old = df2[(df2['landing_page']=='old_page')&(df2['converted']==1)].shape[0]
          convert_new = df2[(df2['landing_page']=='new_page')&(df2['converted']==1)].shape[0]
          n_old = df2[(df2.landing_page == 'old_page')].shape[0]
          n_new= df2[(df2.landing_page == 'new_page')].shape[0]
          print(convert_old,convert_new,n_old,n_new)
```

```
17489 17264 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 [130]: 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.**?

Since z-score (1.31) is less than the critical value (1.96), We fail to reject the null hypothesis. This idea agrees with part j and k findings

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?

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 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 [132]: df3 = df2
```

```
In [133]: df3['intercept']=1
          df3[['control','treatment']] = pd.get_dummies(df3['group'])
          df3[['ab_page', 'old_page']] = pd.get_dummies(df3['landing_page'])
          df3.head()
```

```
/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
  """Entry point for launching an IPython kernel.
/opt/conda/lib/python3.6/site-packages/pandas/core/frame.py:2352: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
  self[k1] = value[k2]
```

Out[133]:

	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	control	treatment	ab_page	old_page
0	1	1	0	0	1
1	1	1	0	0	1
2	1	0	1	1	0
3	1	0	1	1	0
4	1	1	0	0	1

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 [134]: import  statsmodels.api  as sm

          model = sm.Logit(df3['converted'], df3[['intercept', 'ab_page']])
          results = model.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 [135]: results.summary()
```

```
Out[135]: <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:                Sun, 30 Dec 2018   Pseudo R-squ.:               8.077e-06
Time:                        07:42:01   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**?

The P-value associated with ab_page = 0.190 and its greater than 0.05, It differ from the value from Part II becuase its two tailed test where the value from Part II came from one tailed test.

The hypotheses in part II are: * H_0:$p_{new} <= p_{old}$ * H_1:$p_{new} > p_{old}$

The hypotheses associated with the regression model are: * H_0:$p_{new} = p_{old}$ * H_1: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?

Considering other factors such as the user age, language and country probably would lead us to different conclusions. However adding additional terms could lead us to overfitting

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 [136]: countries_df = pd.read_csv('countries.csv')
          countries_df.head()
```

Out[136]:

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

```
In [137]: df4 = countries_df.set_index('user_id').join(df3.set_index('user_id'), how='inner')
          df4.head(1)

Out[137]:         country                   timestamp    group landing_page  converted  \
          user_id
          834778       UK  2017-01-14 23:08:43.304998  control     old_page          0

                   intercept  control  treatment  ab_page  old_page
          user_id
          834778           1        1          0        0         1

In [138]: df4['country'].unique()

Out[138]: array(['UK', 'US', 'CA'], dtype=object)

In [139]: df4[['UK','US','CA']] = pd.get_dummies(df4['country'])
          df4.head(1)

Out[139]:         country                   timestamp    group landing_page  converted  \
          user_id
          834778       UK  2017-01-14 23:08:43.304998  control     old_page          0

                   intercept  control  treatment  ab_page  old_page  UK  US  CA
          user_id
          834778           1        1          0        0         1   0   1   0

In [140]: model2 = sm.Logit(df4['converted'], df4[['intercept','US','UK']])
          results2 = model2.fit()
          results2.summary()

Optimization terminated successfully.
         Current function value: 0.366116
         Iterations 6


Out[140]: <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:                Sun, 30 Dec 2018   Pseudo R-squ.:               1.521e-05
          Time:                        07:42:36   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
US             0.0099      0.013      0.746      0.456      -0.016       0.036
UK            -0.0408      0.027     -1.518      0.129      -0.093       0.012
==============================================================================
"""
```

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 [141]: df4['US_page'] = df4['US'] * df4['ab_page']
          df4['UK_page'] = df4['UK'] * df4['ab_page']
          df4.head(1)
```

Out[141]:

user_id	country	timestamp	group	landing_page	converted	intercept	control	treatment	ab_page	old_page	UK	US	CA	US_page	UK_page
834778	UK	2017-01-14 23:08:43.304998	control	old_page	0	1	1	0	0	1	0	1	0	0	0

```
In [142]: model3 = sm.Logit(df4['converted'], df4[['intercept', 'ab_page', 'US', 'UK', 'US_page'
          results3 = model3.fit()
          results3.summary()

Optimization terminated successfully.
         Current function value: 0.366109
         Iterations 6


Out[142]: <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:                Sun, 30 Dec 2018   Pseudo R-squ.:               3.482e-05
          Time:                        07:42:42   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
ab_page       -0.0206      0.014     -1.505      0.132      -0.047       0.006
US            -0.0057      0.019     -0.306      0.760      -0.043       0.031
UK            -0.0175      0.038     -0.465      0.642      -0.091       0.056
US_page        0.0314      0.027      1.181      0.238      -0.021       0.084
UK_page       -0.0469      0.054     -0.872      0.383      -0.152       0.059
==============================================================================
"""
```

conclusions

```
* We fail to reject the null hypothesis based on the p-values.
* Adding factors such as contries did not have a significant effect on conversion rate.
* Therefore, lunching a new website is not a good idea.
```

0.3 references

```
* udacity
* https://docs.scipy.org/doc/numpy-1.13.0/reference/
* https://www.khanacademy.org/search?page_search_query=logistic%20regression
```

```
In [143]: from subprocess import call
          call(['python', '-m', 'nbconvert', 'Analyze_ab_test_results_notebook.ipynb'])
```

```
Out[143]: 0
```

```
In [ ]:
```