Analyze_ab_test_results_notebook

May 11, 2020

0.1 Analyze A/B Test Results

0.2 Table of Contents

Introduction
For this project, I have generated the A/B test if the new web page significantly is useful more than the old page.
Part I - Probability

```
In [1]: import pandas as pd
        import numpy as np
        import random
        import matplotlib.pyplot as plt
        %matplotlib inline
        #We are setting the seed to assure you get the same answers on quizzes as we set up
        random.seed(42)
```

1. Now, read in the `ab_data.csv` data. Store it in `df`. **Use your dataframe to answer the questions in Quiz 1 of the classroom.**

a. Read in the dataset and take a look at the top few rows here:

```
In [2]: df = pd.read_csv('ab_data.csv')
        df.head()

Out[2]:    user_id                   timestamp      group landing_page  converted
        0   851104  2017-01-21 22:11:48.556739    control     old_page          0
        1   804228  2017-01-12 08:01:45.159739    control     old_page          0
        2   661590  2017-01-11 16:55:06.154213  treatment     new_page          0
        3   853541  2017-01-08 18:28:03.143765  treatment     new_page          0
        4   864975  2017-01-21 01:52:26.210827    control     old_page          1
```

b. Use the cell below to find the number of rows in the dataset.

```
In [3]: df.shape[0]

Out[3]: 294478
```

c. The number of unique users in the dataset.

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

Out[4]: user_id         290584
        timestamp       294478
        group                2
        landing_page         2
        converted            2
        dtype: int64
```

d. The proportion of users converted.

```
In [5]: df['converted'].sum() / df['converted'].count()

Out[5]: 0.11965919355605512
```

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

```
In [6]: df.query('group=="treatment" & landing_page=="old_page"').shape[0] + df.query('group=="c

Out[6]: 3893
```

f. Do any of the rows have missing values?

```
In [7]: df.isnull().sum()

Out[7]: 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 [8]: df2 = df.drop(df[((df['group'] == 'treatment') == (df['landing_page'] == 'new_page')) ==

In [9]: # Double Check all of the correct rows were removed - this should be 0
        df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].sha

Out[9]: 0
```

3. Use **df2** and the cells below to answer questions for **Quiz3** in the classroom.

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

```
In [10]: df2['user_id'].nunique()

Out[10]: 290584
```

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

```
In [11]: sum(df2['user_id'].duplicated())

Out[11]: 1
```

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

```
In [12]: df2[df2['user_id'].duplicated()]

Out[12]:       user_id                   timestamp      group landing_page  converted
         2893   773192  2017-01-14 02:55:59.590927  treatment     new_page          0
```

d. Remove **one** of the rows with a duplicate **user_id**, but keep your dataframe as **df2**.

```
In [13]: df2 = df2.drop(df2[df2['user_id'].duplicated()].index)
         df2.head()

Out[13]:    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
```

4. Use **df2** in the cells below to answer the quiz questions related to **Quiz 4** in the classroom.

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

```
In [14]: df2['converted'].sum() / df2['converted'].count()

Out[14]: 0.11959708724499628
```

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

```
In [15]: df2[df2['group']=="control"]['converted'].sum() / df2[df2['group']=="control"]['convert

Out[15]: 0.1203863045004612
```

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

```
In [16]: df2[df2['group']=="treatment"]['converted'].sum() / df2[df2['group']=="treatment"]['con
```

```
Out[16]: 0.11880806551510564
```

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

```
In [17]: df2[df2['landing_page']=="new_page"].shape[0] / df2['landing_page'].shape[0]
```

```
Out[17]: 0.5000619442226688
```

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.

When the percentage of new_page and old_page people were similar, it is difficult to say that there was a significant difference between the new and old pages.

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.

H0: $p_{new} <= p_{old}$

H1: $p_{new} > p_{old}$

a = 0.05

`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 the H0, p_new = p_old)

```
In [18]: p_new = df2['converted'].mean()
         p_new
```

```
Out[18]: 0.11959708724499628
```

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

```
In [19]: p_old = df2['converted'].mean()
         p_old

Out[19]: 0.11959708724499628
```

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

```
In [20]: n_new = df2[df2['group'] == "treatment"].shape[0]
         n_new

Out[20]: 145310
```

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

```
In [21]: n_old = df2[df2['group'] == "control"].shape[0]
         n_old

Out[21]: 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 [22]: new_page_converted = np.random.binomial(n_new, p_new)/n_new
         new_page_converted

Out[22]: 0.11984722317803317
```

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 [23]: old_page_converted = np.random.binomial(n_old, p_old)/n_old
         old_page_converted

Out[23]: 0.11969795008053746
```

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

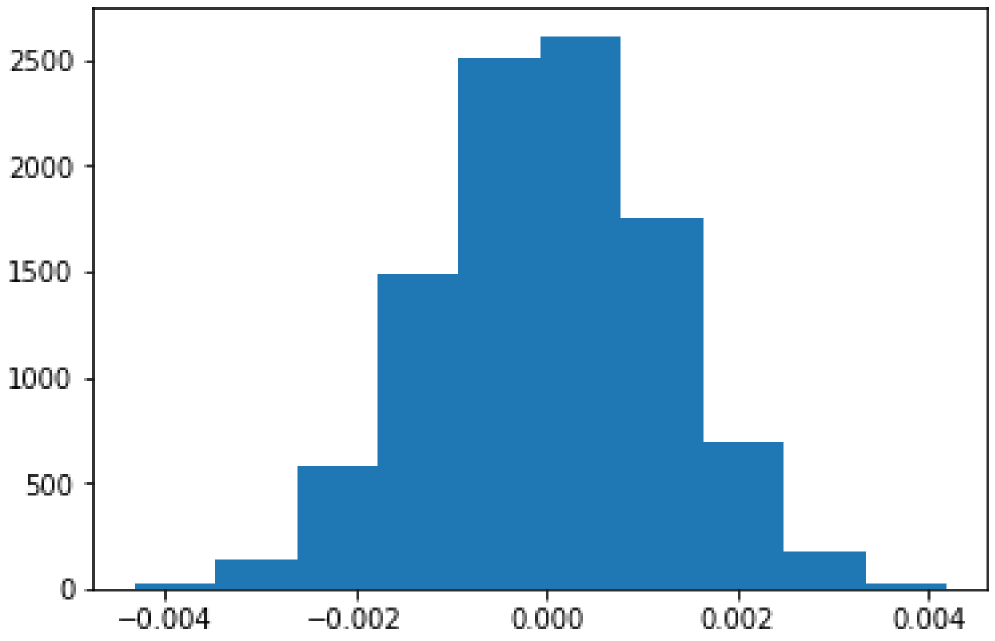
```
In [24]: new_page_converted - old_page_converted

Out[24]: 0.00014927309749571127
```

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 [25]: p_diffs = []
         new_page_mean = np.random.binomial(n_new, p_new, 10000)/n_new
         old_page_mean = np.random.binomial(n_old, p_old, 10000)/n_old
         p_diffs = new_page_mean - old_page_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 [26]: plt.hist(p_diffs);
```



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

```
In [35]: df_new = df2[df2['landing_page']=="new_page"]
         df_old = df2[df2['landing_page']=="old_page"]
         actual_diff = df_new['converted'].mean() - df_old['converted'].mean()
         (p_diffs>actual_diff).mean()

Out[35]: 0.9069000000000004
```

k. Please explain using the vocabulary you've learned in this course what you just computed in part **j.** What is this value called in scientific studies? What does this value mean in terms of whether or not there is a difference between the new and old pages?

The p-value is estimated as 0.905. Therefore, we fail to reject the null hypothesis(H0).

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 [47]: import statsmodels.api as sm

         convert_old = df2[df2['group']=="control"]['converted'].sum()
         convert_new = df2[df2['group']=="treatment"]['converted'].sum()
         n_old = df2[df2['group']=="control"]['converted'].count()
         n_new = df2[df2['group']=="treatment"]['converted'].count()
```

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

Out[48]: (-1.3109241984234394, 0.90505831275902449)
```

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.**?

as the z_score is -1.311, more than half of the conversion rates are same between the old and the new page. The p-value also shows that the old and the new page conversion is similar in 90.5% chance.

The p-value of the z-test agrees that the results are same in parts j. and k.

Part III - A regression approach

`1.` In this final part, you will see that the result you achieved in the A/B test in Part II above can also be achieved by performing regression.

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

Logistic Regression

b. The goal is to use **statsmodels** to fit the regression model you specified in part **a.** to see if there is a significant difference in conversion based on which page a customer receives. However, you first need to create in df2 a column for the intercept, and create a dummy variable column for which page each user received. Add an **intercept** column, as well as an **ab_page** column, which is 1 when an individual receives the **treatment** and 0 if **control**.

```
In [49]: df2['intercept'] = 1
         df2[['ab_page', 'drop']] = pd.get_dummies(df2['landing_page'])
         df2 = df2.drop(['drop'], axis=1)
```

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

```
In [52]: log_mod = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])
         results = log_mod.fit()
         results.summary()
```

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

```
        ---------------------------------------------------------------------------

        AttributeError                            Traceback (most recent call last)

        <ipython-input-52-766103256e5d> in <module>()
          1 log_mod = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])
          2 results = log_mod.fit()
    ----> 3 results.summary()


        /opt/conda/lib/python3.6/site-packages/statsmodels/discrete/discrete_model.py in summary
       2756                 yname_list=None):
       2757         smry = super(BinaryResults, self).summary(yname, xname, title, alpha,
    -> 2758                      yname_list)
       2759         fittedvalues = self.model.cdf(self.fittedvalues)
       2760         absprederror = np.abs(self.model.endog - fittedvalues)


        /opt/conda/lib/python3.6/site-packages/statsmodels/discrete/discrete_model.py in summary
       2548                      ('Log-Likelihood:', None),
       2549                      ('LL-Null:', ["%#8.5g" % self.llnull]),
    -> 2550                      ('LLR p-value:', ["%#6.4g" % self.llr_pvalue])
       2551                      ]
       2552


        /opt/conda/lib/python3.6/site-packages/statsmodels/tools/decorators.py in __get__(self,
         95         if _cachedval is None:
         96             # Call the "fget" function
    ---> 97             _cachedval = self.fget(obj)
         98             # Set the attribute in obj
         99             # print("Setting %s in cache to %s" % (name, _cachedval))


        /opt/conda/lib/python3.6/site-packages/statsmodels/discrete/discrete_model.py in llr_pva
       2403     @cache_readonly
       2404     def llr_pvalue(self):
    -> 2405         return stats.chisqprob(self.llr, self.df_model)
       2406
       2407     @cache_readonly
```

```
        AttributeError: module 'scipy.stats' has no attribute 'chisqprob'
```

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

```
In [53]: 1/np.exp(-0.0150)

Out[53]: 1.0151130646157189
```

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

The old page makes 1.5% more conversion than the new page.
because the Logistic Regression uses the p-value with a two-sided z-score.

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 adding other factors can make the fresh possibility to find the new relationship in this model. However, as the logistic regression could be more complex and contaminated by other new factors, it should be very careful to add more factors.

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 [56]: df_countries = pd.read_csv('countries.csv')
         df3 = pd.merge(df2, df_countries, on='user_id', how='inner')
         df3[['CA', 'UK', 'US']] = pd.get_dummies(df3['country'])

         Log_mod = sm.Logit(df3['converted'], df3[['intercept', 'ab_page', 'CA', 'UK']])
         results = Log_mod.fit()
         results.summary()

Optimization terminated successfully.
         Current function value: 0.366113
         Iterations 6


        ---------------------------------------------------------------------------
```

```
    AttributeError                            Traceback (most recent call last)

    <ipython-input-56-66faf203d961> in <module>()
      5 Log_mod = sm.Logit(df3['converted'], df3[['intercept', 'ab_page', 'CA', 'UK']])
      6 results = Log_mod.fit()
----> 7 results.summary()


    /opt/conda/lib/python3.6/site-packages/statsmodels/discrete/discrete_model.py in summary
   2756                  yname_list=None):
   2757         smry = super(BinaryResults, self).summary(yname, xname, title, alpha,
-> 2758                      yname_list)
   2759         fittedvalues = self.model.cdf(self.fittedvalues)
   2760         abspredetrror = np.abs(self.model.endog - fittedvalues)


    /opt/conda/lib/python3.6/site-packages/statsmodels/discrete/discrete_model.py in summary
   2548                      ('Log-Likelihood:', None),
   2549                      ('LL-Null:', ["%#8.5g" % self.llnull]),
-> 2550                      ('LLR p-value:', ["%#6.4g" % self.llr_pvalue])
   2551                      ]
   2552


    /opt/conda/lib/python3.6/site-packages/statsmodels/tools/decorators.py in __get__(self,
     95         if _cachedval is None:
     96             # Call the "fget" function
---> 97             _cachedval = self.fget(obj)
     98             # Set the attribute in obj
     99             # print("Setting %s in cache to %s" % (name, _cachedval))


    /opt/conda/lib/python3.6/site-packages/statsmodels/discrete/discrete_model.py in llr_pva
   2403     @cache_readonly
   2404     def llr_pvalue(self):
-> 2405         return stats.chisqprob(self.llr, self.df_model)
   2406
   2407     @cache_readonly


    AttributeError: module 'scipy.stats' has no attribute 'chisqprob'
```

When I measured the p-value of the each countries, it was difficult to find a significant p-value difference among the countries. For instance, setting US as the baseline, the p-value of Canada was still greater than 0.1.

Therefore, it is a bit complex to divide the countries in my opinion.

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 [62]: df3['CA_page'] = df3['ab_page']*df3['CA']
         df3['UK_page'] = df3['ab_page']*df3['UK']
         df3['US_page'] = df3['ab_page']*df3['US']

         logit_mod = sm.Logit(df3['converted'], df3[['intercept', 'ab_page', 'CA', 'UK', 'CA_pag
         results = logit_mod.fit()
         results.summary()
```

```
Optimization terminated successfully.
         Current function value: 0.366109
         Iterations 7



        ---------------------------------------------------------------------------

        AttributeError                            Traceback (most recent call last)

        <ipython-input-62-48cfbb1cb2e1> in <module>()
          5 logit_mod = sm.Logit(df3['converted'], df3[['intercept', 'ab_page', 'CA', 'UK', 'CA_
          6 results = logit_mod.fit()
    ----> 7 results.summary()


        /opt/conda/lib/python3.6/site-packages/statsmodels/discrete/discrete_model.py in summary
       2756                 yname_list=None):
       2757         smry = super(BinaryResults, self).summary(yname, xname, title, alpha,
    -> 2758                      yname_list)
       2759         fittedvalues = self.model.cdf(self.fittedvalues)
       2760         absprederror = np.abs(self.model.endog - fittedvalues)


        /opt/conda/lib/python3.6/site-packages/statsmodels/discrete/discrete_model.py in summary
       2548                      ('Log-Likelihood:', None),
       2549                      ('LL-Null:', ["%#8.5g" % self.llnull]),
    -> 2550                      ('LLR p-value:', ["%#6.4g" % self.llr_pvalue])
       2551                      ]
       2552


        /opt/conda/lib/python3.6/site-packages/statsmodels/tools/decorators.py in __get__(self,
         95         if _cachedval is None:
         96             # Call the "fget" function
    ---> 97             _cachedval = self.fget(obj)
```

```
     98             # Set the attribute in obj
     99             # print("Setting %s in cache to %s" % (name, _cachedval))


    /opt/conda/lib/python3.6/site-packages/statsmodels/discrete/discrete_model.py in llr_pva
   2403     @cache_readonly
   2404     def llr_pvalue(self):
-> 2405         return stats.chisqprob(self.llr, self.df_model)
   2406
   2407     @cache_readonly


    AttributeError: module 'scipy.stats' has no attribute 'chisqprob'
```

When we consider about the z-scores of the values, it is not so useful to use this complex regression model.

```
In [63]: from subprocess import call
         call(['python', '-m', 'nbconvert', 'Analyze_ab_test_results_notebook.ipynb'])

Out[63]: 0

In [ ]:
```