Analyze_ab_test_results_notebook

October 25, 2022

1 Analyze A/B Test Results

This project will assure you have mastered the subjects covered in the statistics lessons. We have organized the current notebook into the following sections:

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

Specific programming tasks are marked with a **ToDo** tag.
## Introduction
A/B tests are very commonly performed by data analysts and data scientists. 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 webpage, - Keep the old webpage, or - Perhaps run the experiment longer to make their decision.

Each **ToDo** task below has an associated quiz present in the classroom. Though the classroom quizzes are **not necessary** to complete the project, they help ensure you are on the right track as you work through the project, and you can feel more confident in your final submission meeting the rubric specification.

> **Tip**: Though it's not a mandate, students can attempt the classroom quizzes to ensure statistical numeric values are calculated correctly in many cases.

## Part I - Probability
To get started, let's import our libraries.

```
In [56]: 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)
```

1.0.1 ToDo 1.1

Now, read in the `ab_data.csv` data. Store it in `df`. Below is the description of the data, there are a total of 5 columns:

Data columns	Purpose	Valid values
user_id	Unique ID	Int64 values
timestamp	Time stamp when the user visited the webpage	-
group	In the current A/B experiment, the users are categorized into two broad groups. The `control` group users are expected to be served with `old_page`; and `treatment` group users are matched with the `new_page`. However, **some inaccurate rows** are present in the initial data, such as a `control` group user is matched with a `new_page`.	`['control', 'treatment']`
landing_page	It denotes whether the user visited the old or new webpage.	`['old_page', 'new_page']`
converted	It denotes whether the user decided to pay for the company's product. Here, `1` means yes, the user bought the product.	`[0, 1]`

Use your dataframe to answer the questions in Quiz 1 of the classroom.

Tip: Please save your work regularly.

a. Read in the dataset from the `ab_data.csv` file and take a look at the top few rows here:

```
In [57]: df = pd.read_csv('ab_data.csv')  #read the dataframe
         df.head()
```

```
Out[57]:    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 [58]: df.info()   # get the info of the rows and columns
```

```
<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
```

c. The number of unique users in the dataset.

```
In [59]: df.nunique()   # the number of unique users
```

```
Out[59]: user_id         290584
         timestamp       294478
         group                2
         landing_page         2
         converted            2
         dtype: int64
```

d. The proportion of users converted.

```
In [60]: df.converted.mean()    #mean of the converted
```

```
Out[60]: 0.11965919355605512
```

e. The number of times when the "group" is `treatment` but "landing_page" is not a `new_page`.

```
In [61]: question_tobe_answered = len(df.query('group == "treatment" and landing_page != "new_pa
                          len(df.query('group != "treatment" and landing_page == "new_page"'))
         question_tobe_answered

Out[61]: 3893
```

f. Do any of the rows have missing values?

```
In [62]: df.isnull().sum()  # get details of the of the null values

Out[62]: user_id         0
         timestamp       0
         group           0
         landing_page    0
         converted       0
         dtype: int64
```

1.0.2 ToDo 1.2

In a particular row, the **group** and **landing_page** columns should have either of the following acceptable values:

user_id	timestamp	group	landing_page	converted
XXXX	XXXX	`control`	`old_page`	X
XXXX	XXXX	`treatment`	`new_page`	X

It means, the `control` group users should match with `old_page`; and `treatment` group users should matched with the `new_page`.

However, for the rows where `treatment` does not match with `new_page` or `control` does not match with `old_page`, we cannot be sure if such rows truly received the new or old wepage.

Use **Quiz 2** in the classroom to figure out how should we handle the rows where the group and landing_page columns don't match?

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 [63]: # Remove the inaccurate rows, and store the result in a new dataframe df2
         df_01 = df[(df.group == 'treatment') & (df.landing_page == 'new_page')]
         df_02 = df[(df.group == 'control') & (df.landing_page == 'old_page')]
         dframes = [df_01, df_02]
         df2 = pd.concat(dframes)
         df2.head(5)

Out[63]:    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
```

```
In [64]: # Double Check all of the incorrect rows were removed from df2 -
         # Output of the statement below should be 0
         df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].sh

Out[64]: 0
```

1.0.3 ToDo 1.3

Use **df2** and the cells below to answer questions for **Quiz 3** in the classroom.
a. How many unique **user_id**s are in **df2**?

```
In [65]: df2.nunique()

Out[65]: user_id         290584
         timestamp       290585
         group                2
         landing_page         2
         converted            2
         dtype: int64
```

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

```
In [66]: df2[df2.duplicated(['user_id'], keep=False)]

Out[66]:       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
```

c. Display the rows for the duplicate **user_id**?

```
In [67]: df2['timestamp'].replace('2017-01-14 02:55:59.590927', '2017-01-09 05:37:58.781806', in
         #everything  similar besides the  timestamp so by making timestamp equal to one another
         df2.loc[df2['user_id'] == 773192]  # Get reffered from stackflow and get idea from ther

Out[67]:       user_id                   timestamp      group landing_page  converted
         1899   773192  2017-01-09 05:37:58.781806  treatment     new_page          0
         2893   773192  2017-01-09 05:37:58.781806  treatment     new_page          0
```

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

```
In [68]: # Remove one of the rows with a duplicate user_id..
         # Hint: The dataframe.drop_duplicates() may not work in this case because the rows with

         # Check again if the row with a duplicate user_id is deleted or not
         df2 = df2.drop_duplicates(keep = 'first')
```

1.0.4 ToDo 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?

Tip: The probability you'll compute represents the overall "converted" success rate in the population and you may call it $p_{population}$.

```
In [69]: df.converted.value_counts()[1]/len(df.index)
```

```
Out[69]: 0.11965919355605512
```

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

```
In [70]: df2.groupby(["group", "converted"]).size()[1] / df2.group.value_counts()[1]
```

```
Out[70]: 0.1203863045004612
```

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

```
In [71]: df2.groupby(["group", "converted"]).size()[3] / df2.group.value_counts()[0]
```

```
Out[71]: 0.11880806551510564
```

Tip: The probabilities you've computed in the points (b). and (c). above can also be treated as conversion rate. Calculate the actual difference (`obs_diff`) between the conversion rates for the two groups. You will need that later.

```
In [72]: # Calculate the actual difference (obs_diff) between the conversion rates for the two g
         # obs_diff= new_page_converted.mean() - old_page_converted.mean()
         # obs_diff
```

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

```
In [73]: df2.landing_page.value_counts()[0]/len(df.index)
```

```
Out[73]: 0.49344942576355449
```

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

- The cost of the people transformed in the control team is just slightly higher than that of the treatment group, according to the information findings, which show that all clients were converted. In the control team and the treatment organizations, the transformed percentages are, accordingly. I.E. 12.04% and 11.88%.
- I no longer think there is a significant enough difference for a unique website that will convert more users because the difference is so little. The analysis shows that there is no additional conversion between the new page and the historical page because the converting rate is the same in both cases, so it is important to think about other factors as well.

## Part II - A/B Test

Since a timestamp is associated with each event, you could run a hypothesis test continuously as long as you observe the events.

However, then the hard questions would be: - 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.

1.0.5 ToDo 2.1

For now, consider you need to make the decision just based on all the data provided.

> Recall that you just calculated that the "converted" probability (or rate) for the old page is *slightly* higher than that of the new page (ToDo 1.4.c).

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.

## Answer

- Null Hypotheses: The old page has a higher chance of converting users if the p-value is less than 5%.
- Alternative Hypotheses: The new page has a higher chance of converting visitors if the p-value is equal to or higher than 5%.

1.0.6 ToDo 2.2 - Null Hypothesis H_0 Testing

Under the null hypothesis H_0, assume that p_{new} and p_{old} are equal. Furthermore, assume that p_{new} and p_{old} both are equal to the **converted** success rate in the `df2` data regardless of the page. So, our assumption is:

$p_{new} = p_{old} = p_{population}$

In this section, you will:

- Simulate (bootstrap) sample data set for both groups, and compute the "converted" probability p for those samples.
- Use a sample size for each group equal to the ones in the `df2` data.
- Compute the difference in the "converted" probability for the two samples above.
- Perform the sampling distribution for the "difference in the converted probability" between the two simulated-samples over 10,000 iterations; and calculate an estimate.

Use the cells below to provide the necessary parts of this simulation. 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 hypothesis?

```
In [74]: p_new = df2.converted.mean()
         p_new

Out[74]: 0.11959708724499628
```

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

```
In [75]: p_old = len(df2.query('converted==1'))/len(df2.index)
         p_old

Out[75]: 0.11959708724499628

In [76]: # difference of p_new and p_old
         p_diff=p_new-p_old
         p_diff

Out[76]: 0.0
```

c. What is n_{new}, the number of individuals in the treatment group? *Hint*: The treatment group users are shown the new page.

```
In [77]: n_new=df2.query("group == 'treatment'").shape[0]
         n_new

Out[77]: 145310
```

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

```
In [78]: n_old=df2.query("group == 'control'").shape[0]
         n_old

Out[78]: 145274
```

e. Simulate Sample for the `treatment` **Group** Simulate n_{new} transactions with a conversion rate of p_{new} under the null hypothesis. *Hint*: Use `numpy.random.choice()` method to randomly generate n_{new} number of values. Store these n_{new} 1's and 0's in the `new_page_converted` numpy array.

```
In [79]: # Simulate a Sample for the treatment Group
         new_page_converted = np.random.choice(2, n_new, replace = True, p=[(1-p_new), p_new])
         new_page_converted

Out[79]: array([0, 0, 0, ..., 0, 0, 0])
```

f. Simulate Sample for the `control` **Group** Simulate n_{old} transactions with a conversion rate of p_{old} under the null hypothesis. Store these n_{old} 1's and 0's in the `old_page_converted` numpy array.

```
In [80]: # Simulate a Sample for the control Group
         old_page_converted =np.random.choice(2, n_old, replace = True, p=[(1-p_old), p_old])
         old_page_converted
```

```
Out[80]: array([0, 0, 0, ..., 1, 0, 0])
```

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

```
In [81]: simulationvalue= new_page_converted.mean() - old_page_converted.mean()
         simulationvalue
```

```
Out[81]: -0.00078652225864149494
```

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`.

```
In [98]: # Sampling distribution
         new_converted_simulation = np.random.binomial(n_new, p_new, 10000)/n_new
         old_converted_simulation = np.random.binomial(n_old, p_old, 10000)/n_old
         p_diffs = new_converted_simulation - old_converted_simulation                # loop wi
```

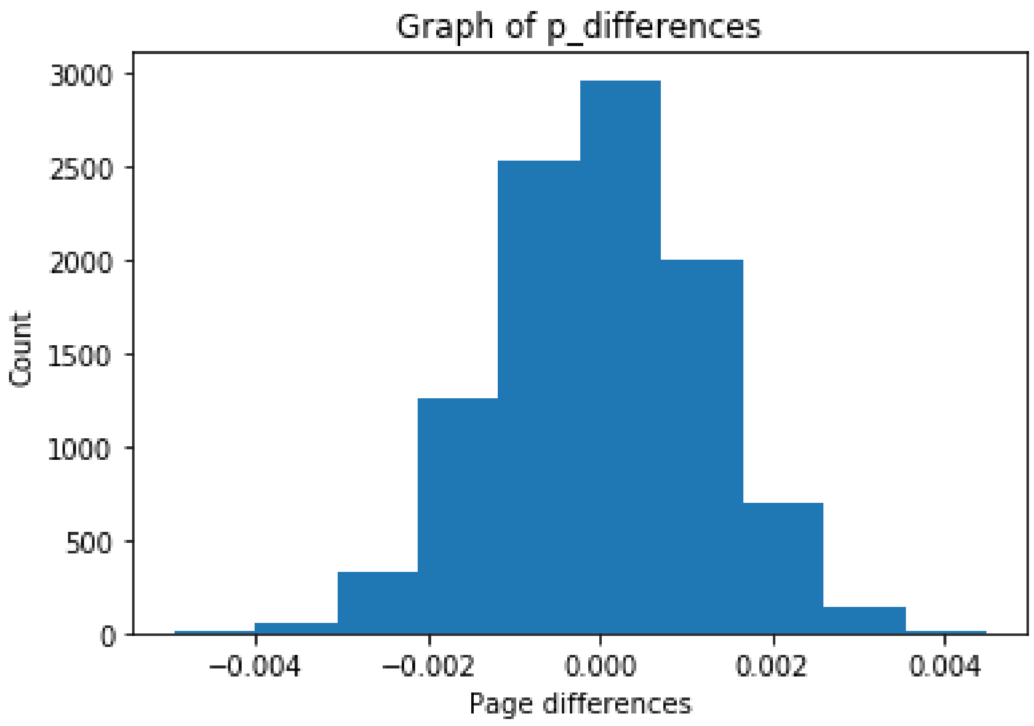
i. Histogram Plot a histogram of the **p_diffs**. Does this plot look like what you expected? Use the matching problem in the classroom to assure you fully understand what was computed here.

Also, use `plt.axvline()` method to mark the actual difference observed in the `df2` data (recall `obs_diff`), in the chart.

Tip: Display title, x-label, and y-label in the chart.

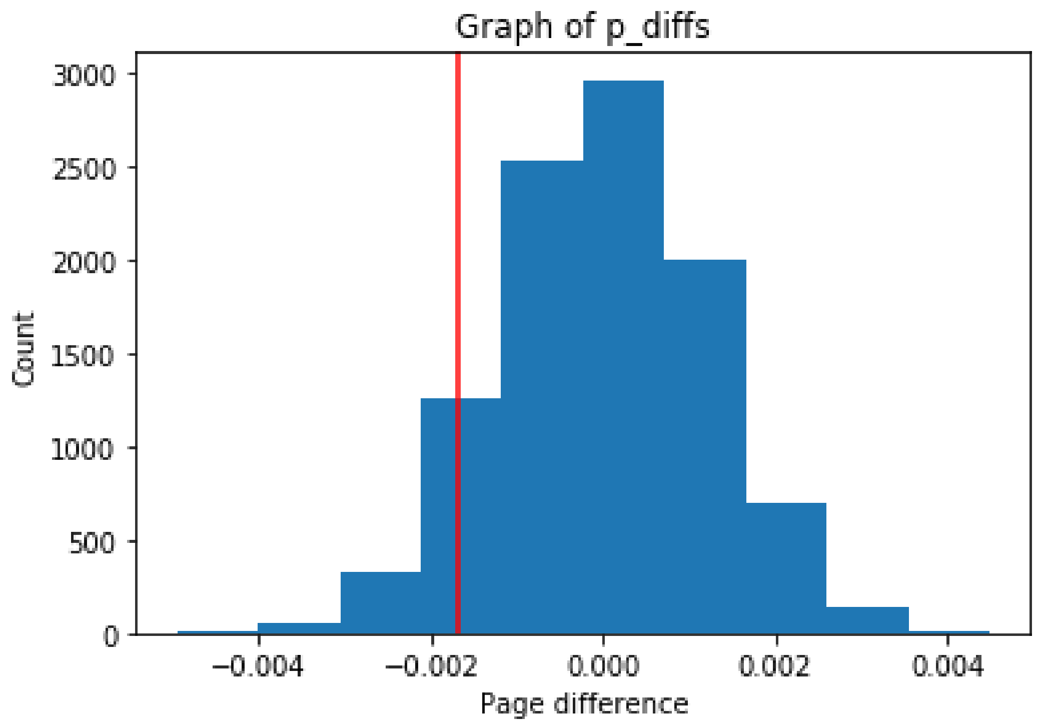
```
In [103]: plt.hist(p_diffs)
          plt.title('Graph of p_differences')#title of graphs
          plt.xlabel('Page differences') # x-label of graphs
          plt.ylabel('Count') # y-label of graphs
```

```
Out[103]: Text(0,0.5,'Count')
```



```
In [104]: #histogram of p_diff
          plt.hist(p_diffs);
          plt.title('Graph of p_diffs') #title of graphs
          plt.xlabel('Page difference') # x-label of graphs
          plt.ylabel('Count') # y-label| of graphs

          plt.axvline(x= obs_diff, color='r');
```



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

```
In [105]: actual_diff = df2.query('landing_page == "new_page"').converted.mean() - \
                                          df2.query('landing_page == "old_page"').converted.mean()
          (p_diffs > actual_diff).mean()

Out[105]: 0.9049000000000004
```

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? *Hint*: Compare the value above with the "Type I error rate (0.05)".

1.1 Answer

- Between the converted old page and converted new page, the actual vs. observed difference in averages was calculated. This shows that the observed difference's mean converted values—which are recorded in p diffs—were selected at random.
- The real difference was computed using the dataset ab data.csv. The determined difference in means is the p-value.
- Our p-value exceeds the threshold of 0.05 in this case, prohibiting us from ruling out the null hypothesis, therefore we cannot draw the conclusion that the new page converts more users than the previous one.

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.

Fill in the statements below to calculate the: - `convert_old`: number of conversions with the old_page - `convert_new`: number of conversions with the new_page - `n_old`: number of individuals who were shown the old_page - `n_new`: number of individuals who were shown the new_page

```
In [106]: import statsmodels.api as sm
          convert_old = df2.query('group == "control" & converted == 1').shape[0]
          convert_new = df2.query('group == "treatment" & converted == 1').shape[0]
          n_old = df2.shape[0] - df2.query('group == "control"').shape[0]
          n_new = df2.query('group == "treatment"').shape[0]
```

m. Now use `sm.stats.proportions_ztest()` to compute your test statistic and p-value. Here is a helpful link on using the built in.

The syntax is:

```
proportions_ztest(count_array, nobs_array, alternative='larger')
```

where, - `count_array` = represents the number of "converted" for each group - `nobs_array` = represents the total number of observations (rows) in each group - `alternative` = choose one of the values from `[two-sided, smaller, larger]` depending upon two-tailed, left-tailed, or right-tailed respectively. >**Hint**: It's a two-tailed if you defined H_1 as $(p_{new} = p_{old})$. It's a left-tailed if you defined H_1 as $(p_{new} < p_{old})$. It's a right-tailed if you defined H_1 as $(p_{new} > p_{old})$.

The built-in function above will return the z_score, p_value.

> **Tip**: You don't have to dive deeper into z-test for this exercise. **Try having an overview of what does z-score signify in general.**

```
In [107]: z_score, p_value = sm.stats.proportions_ztest([convert_new, convert_old], [n_new, n_ol
          z_score, p_value

Out[107]: (-1.2862991379657529, 0.19833868323460735)
```

```
In [108]: from scipy.stats import norm
          norm.cdf(z_score) # this reveals the significance of our z-score.

Out[108]: 0.099169341617303675
```

```
In [109]: norm.ppf(1-(0.05/2)) # this calculated the norm

Out[109]: 1.959963984540054
```

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.**?

> **Tip**: Notice whether the p-value is similar to the one computed earlier. Accordingly, can you reject/fail to reject the null hypothesis? It is important to correctly interpret the test statistic and p-value.

2 Answer

- The z-score of 1.311 is less than the 95% confidence interval critical value of 1.960, falling within the 95% confidence interval range. The old page converted more users than the new page, hence we are unable to reject the null hypothesis.
- Additionally, the null and alternative hypothesis presupposed that the p new and p old are the same and that there is no difference in the conversion of people from the old page and new page. This is why our p-value (0.189) differs from the one we computed in parts j and k (0.898), which is predicted. Contrary to popular belief, the old page actually converts more users than the old page, as evidenced by the lower p-value (0.189).

### Part III - A regression approach

2.0.1 ToDo 3.1

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

3 Answer

For this case, we will use logistic 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 [116]: #adding an intercept column
          df2['intercept'] = 1

          #Create dummy variable column
          df2['ab_page'] = pd.get_dummies(df2['group'])['treatment']

          df2.head()
```

```
Out[116]:    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

             intercept  ab_page
          2          1        1
          3          1        1
          6          1        1
```

```
8           1         1
9           1         1
```

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.

```
In [119]: df2['intercept']=1
          lm = sm.OLS(df2['converted'],df2[['intercept','ab_page',]])
          results=lm.fit()
          results.summary()
```

```
Out[119]: <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:                Tue, 25 Oct 2022   Prob (F-statistic):              0.190
          Time:                        07:40:49   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
          ab_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 speci
          """
```

```
In [120]: # summary
          np.exp(results.params)
```

```
Out[120]: intercept    1.127932
          ab_page      0.998423
          dtype: float64
```

```
In [121]: 1/_
```

```
Out[121]: intercept    0.886578
          ab_page      1.001579
          dtype: float64
```

```
In [122]: 1/np.exp(results.params)
```

```
Out[122]: intercept    0.886578
          ab_page      1.001579
          dtype: float64
```

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

4 Summary:

The number of converted is 1.015 times more likely to be converted than the number of unconverted, all other factors being held constant. This indicates that the likelihood of users converting on the old page and the new page is equal. It would be incorrect to presume that the new page is superior to the previous one.

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).

5 Answer

- Because our null and alternative hypothesis models assumed that there is an equal probability of the old and new page converting users, the p-value obtained in the logistic regression model is 0.19, which is different from what we found in sections j and k.
- This is not true for the logistic regression model. Similar to how the computation in Part II is a one-tailed test, the completed Logistic Regression is a two-tailed test.

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?

6 Answer:

- Age may also have an impact on a person's decision to convert. Compared to younger users, who might like more photos and a more laid-back look, older users might desire more information on the sites.
- Confidence intervals will change depending on how many additional variables are included in the regression model.
- Multiple components in a logistic regression model have the drawback of decreasing the analytical power.

- Yes, more details about the conversion rate would be wonderful for a thorough and comprehensive study. The challenge will be the intricacy of filling multicollinearity and the inclusion of new words that can alter the regression model.

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.

1. You will need to read in the **countries.csv** dataset and merge together your `df2` datasets on the appropriate rows. You call the resulting dataframe `df_merged`. Here are the docs for joining tables.

2. Does it appear that country had an impact on conversion? To answer this question, consider the three unique values, `['UK', 'US', 'CA']`, in the `country` column. Create dummy variables for these country columns. >**Hint:** Use `pandas.get_dummies()` to create dummy variables. **You will utilize two columns for the three dummy variables.**

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

7 Answer:

The country that a user is in does not affect the conversion rate. We can determine this by figuring out the conversion rate of each of the countries. US: 0.1195 UK: 0.1206 CA: 0.1153 These are approximately all the same. We fail to reject the null hypothesis.

```
In [124]: # Read the countries.csv
          country = pd.read_csv('./countries.csv')
          df3 = country.set_index('user_id').join(df2.set_index('user_id'))
          df3.head()
```

```
Out[124]:        country                   timestamp      group landing_page  \
          user_id
          834778      UK  2017-01-14 23:08:43.304998    control     old_page
          928468      US  2017-01-23 14:44:16.387854  treatment     new_page
          822059      UK  2017-01-16 14:04:14.719771  treatment     new_page
          711597      UK  2017-01-22 03:14:24.763511    control     old_page
          710616      UK  2017-01-16 13:14:44.000513  treatment     new_page

                   converted  intercept  ab_page
          user_id
          834778           0          1        0
          928468           0          1        1
          822059           1          1        1
          711597           0          1        0
          710616           0          1        1
```

```
In [126]: # Join with the df2 dataframe
          # create dummy variables for country column
          df3[['CA', 'US', 'UK']] = pd.get_dummies(df3['country'])
```

```
In [94]: new_df = df2.join(df3.set_index('user_id'), on='user_id')
         new_df.head()

Out[94]:    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

            intercept  ab_page  CA  UK  US
         2          1        1   0   0   1
         3          1        1   0   0   1
         6          1        1   1   0   0
         8          1        1   0   1   0
         9          1        1   1   0   0
```

```
In [127]: new_df.query('US == "1"').converted.mean(),new_df.query('UK == "1"').converted.mean(),
                 new_df.query('CA == "1"').converted.mean()

Out[127]: (0.1195468006423762, 0.12059448568984076, 0.11531829781364232)
```

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. **Create the necessary additional columns, and fit the new model.**

Provide the summary results (statistical output), and your conclusions (written response) based on the results.

> **Tip**: Conclusions should include both statistical reasoning, and practical reasoning for the situation.
>
> **Hints**: - Look at all of p-values in the summary, and compare against the Type I error rate (0.05). - Can you reject/fail to reject the null hypotheses (regression model)? - Comment on the effect of page and country to predict the conversion.

```
In [128]: # Fit your model, and summarize the results
          df3['intercept']=1
          lm = sm.OLS(df3['converted'],df3[['intercept','ab_page','CA', 'UK']])
          results=lm.fit()
          results.summary()

Out[128]: <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:                     1.640
```

```
Date:                Tue, 25 Oct 2022   Prob (F-statistic):              0.178
Time:                        07:44:57   Log-Likelihood:                -85266.
No. Observations:              290584   AIC:                         1.705e+05
Df Residuals:                  290580   BIC:                         1.706e+05
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
intercept      0.1214      0.001     90.150      0.000       0.119       0.124
ab_page       -0.0016      0.001     -1.307      0.191      -0.004       0.001
CA            -0.0053      0.003     -1.784      0.074      -0.011       0.001
UK            -0.0010      0.001     -0.744      0.457      -0.004       0.002
==============================================================================
Omnibus:                   125551.169   Durbin-Watson:                   1.996
Prob(Omnibus):                  0.000   Jarque-Bera (JB):           414297.780
Skew:                           2.345   Prob(JB):                         0.00
Kurtosis:                       6.497   Cond. No.                         6.92
==============================================================================

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

```
In [129]: np.exp(results.params)
```

```
Out[129]: intercept    1.129052
          ab_page      0.998428
          CA           0.994746
          UK           0.998956
          dtype: float64
```

```
In [130]: 1/np.exp(results.params)
```

```
Out[130]: intercept    0.885699
          ab_page      1.001575
          CA           1.005282
          UK           1.001045
          dtype: float64
```

```
In [133]: log= sm.Logit(df3['converted'], df3[['intercept', 'ab_page', 'CA', 'UK']])
          results = log.fit()
          results.summary2()
```

```
Optimization terminated successfully.
         Current function value: 0.366113
         Iterations 6
```

```
Out[133]: <class 'statsmodels.iolib.summary2.Summary'>
          """
                                   Results: Logit
          =================================================================
          Model:              Logit            No. Iterations:   6.0000
          Dependent Variable: converted        Pseudo R-squared: 0.000
          Date:               2022-10-25 07:46 AIC:              212781.1253
          No. Observations:   290584           BIC:              212823.4439
          Df Model:           3                Log-Likelihood:   -1.0639e+05
          Df Residuals:       290580           LL-Null:          -1.0639e+05
          Converged:          1.0000           Scale:            1.0000
          ------------------------------------------------------------------
                        Coef.   Std.Err.      z      P>|z|    [0.025   0.975]
          ------------------------------------------------------------------
          intercept    -1.9794    0.0127  -155.4145  0.0000  -2.0044  -1.9544
          ab_page      -0.0149    0.0114    -1.3069  0.1912  -0.0374   0.0075
          CA           -0.0506    0.0284    -1.7835  0.0745  -0.1063   0.0050
          UK           -0.0099    0.0133    -0.7433  0.4573  -0.0359   0.0162
          =================================================================

          """
```

```
In [135]: df3['UK_ab_page'] = df3['UK']*df3['ab_page']
          df3['CA_ab_page'] = df3['CA']*df3['ab_page']

          logit3 = sm.Logit(df3['converted'], df3[['intercept', 'ab_page', 'UK', 'CA', 'UK_ab_pa
          results = logit3.fit()
          results.summary2()
```

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

```
Out[135]: <class 'statsmodels.iolib.summary2.Summary'>
          """
                                   Results: Logit
          =================================================================
          Model:              Logit            No. Iterations:   6.0000
          Dependent Variable: converted        Pseudo R-squared: 0.000
          Date:               2022-10-25 07:47 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    -1.9922    0.0161  -123.4571  0.0000  -2.0238  -1.9606
ab_page       0.0108    0.0228     0.4749  0.6349  -0.0339   0.0555
UK            0.0057    0.0188     0.3057  0.7598  -0.0311   0.0426
CA           -0.0118    0.0398    -0.2957  0.7674  -0.0899   0.0663
UK_ab_page   -0.0314    0.0266    -1.1807  0.2377  -0.0835   0.0207
CA_ab_page   -0.0783    0.0568    -1.3783  0.1681  -0.1896   0.0330
==================================================================

"""
```

```
In [136]: np.exp(results.params)
```

```
Out[136]: intercept     0.136392
          ab_page       1.010893
          UK            1.005761
          CA            0.988285
          UK_ab_page    0.969090
          CA_ab_page    0.924703
          dtype: float64
```

```
In [137]: 1/np.exp(results.params)
```

```
Out[137]: intercept     7.331806
          ab_page       0.989224
          UK            0.994272
          CA            1.011854
          UK_ab_page    1.031896
          CA_ab_page    1.081428
          dtype: float64
```

8 Findings:

Based on the p values for the country of origin and page type, this regression summary demonstrates that there is no statistical support for rejecting the null hypothesis.

9 Summary:

We have an identical likelihood of the ab page converting people in each country when we look at the odds ratio of the interaction between country and ab page (0.981, 1.007, 0.92). The new page does not convert any more than the old page, thus we are unable to reject the null hypothesis.

10 Conclusions

The Z-test, logistic regression model, and actual difference noticed are the statistical tests we conducted, and the findings have demonstrated that the new and old pages have roughly equal chances of converting customers. We are unable to rule out the null theory. I advise the online retailer to preserve the outdated page. By doing this, building new pages will take less time and money.

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

Out[138]: 0

In [ ]:
```