Analyze_ab_test_results_notebook

May 21, 2020

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 [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]: # reading the data
        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]: # using shape to find the number of rows
        df.shape
```

```
Out[3]: (294478, 5)
```

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
```

```
In [5]: df.user_id.value_counts()
```

```
Out[5]: 637561    2
        821876    2
        643869    2
        938802    2
        916765    2
        690255    2
        737500    2
        680018    2
        853835    2
        736746    2
        722827    2
        904340    2
        757485    2
        863300    2
```

```
905507    2
902109    2
782432    2
644294    2
899374    2
881704    2
656951    2
869729    2
720460    2
889529    2
812376    2
846972    2
776770    2
859842    2
844475    2
848746    2
          ..
874753    1
868610    1
870659    1
880900    1
876806    1
669933    1
878855    1
856328    1
858377    1
852234    1
854283    1
864524    1
696574    1
702717    1
700668    1
690427    1
688378    1
694521    1
692472    1
714999    1
712950    1
719093    1
717044    1
706803    1
704754    1
710897    1
708848    1
665839    1
663790    1
630836    1
Name: user_id, Length: 290584, dtype: int64
```

d. The proportion of users converted.

```
In [6]: df.converted.mean()
```

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

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

```
In [7]: df.query('group=="treatment" and landing_page != "new_page"')
```

```
Out[7]:
```

	user_id	timestamp	group	landing_page	converted
308	857184	2017-01-20 07:34:59.832626	treatment	old_page	0
327	686623	2017-01-09 14:26:40.734775	treatment	old_page	0
357	856078	2017-01-12 12:29:30.354835	treatment	old_page	0
685	666385	2017-01-23 08:11:54.823806	treatment	old_page	0
713	748761	2017-01-10 15:47:44.445196	treatment	old_page	0
776	820951	2017-01-04 02:42:54.770627	treatment	old_page	0
889	839954	2017-01-06 20:58:22.280929	treatment	old_page	0
1037	880442	2017-01-07 21:42:39.026815	treatment	old_page	0
1106	817911	2017-01-17 21:51:43.220160	treatment	old_page	0
1376	844475	2017-01-20 14:25:37.359614	treatment	old_page	0
1551	838336	2017-01-14 22:05:24.310302	treatment	old_page	0
1706	916207	2017-01-20 11:53:39.683012	treatment	old_page	0
1762	690127	2017-01-11 16:02:57.551297	treatment	old_page	1
2233	869707	2017-01-02 18:36:28.222510	treatment	old_page	0
2422	853156	2017-01-15 23:19:45.427866	treatment	old_page	0
2689	793494	2017-01-09 02:09:08.534282	treatment	old_page	0
3262	710871	2017-01-15 13:58:39.846106	treatment	old_page	0
3306	809229	2017-01-17 22:37:26.403828	treatment	old_page	0
3364	915093	2017-01-16 18:02:59.006193	treatment	old_page	0
3689	878413	2017-01-03 13:41:19.090123	treatment	old_page	0
3869	792890	2017-01-12 21:42:36.159299	treatment	old_page	0
4000	706721	2017-01-04 00:32:24.564711	treatment	old_page	0
4043	846754	2017-01-24 01:27:40.512402	treatment	old_page	0
4074	768200	2017-01-21 15:48:44.216867	treatment	old_page	0
4475	706878	2017-01-09 20:33:39.727111	treatment	old_page	0
4537	761716	2017-01-23 20:32:13.298444	treatment	old_page	0
4961	844946	2017-01-04 07:20:58.924520	treatment	old_page	1
5418	926559	2017-01-16 00:59:10.283392	treatment	old_page	0
5492	662456	2017-01-07 19:48:48.540429	treatment	old_page	0
5800	709280	2017-01-19 22:05:06.906174	treatment	old_page	1
...	...	...	...	...	...
288375	631156	2017-01-04 03:05:13.816388	treatment	old_page	0
288465	767964	2017-01-19 09:41:32.875795	treatment	old_page	1
289242	698366	2017-01-04 00:22:43.306653	treatment	old_page	0
289665	693835	2017-01-20 11:44:50.517253	treatment	old_page	0
289799	909162	2017-01-09 17:12:38.910965	treatment	old_page	0
289846	934943	2017-01-04 18:45:15.921776	treatment	old_page	0
290062	928175	2017-01-05 03:51:08.933502	treatment	old_page	1

```
290149   858910  2017-01-10 05:20:37.997730  treatment     old_page          1
290328   658911  2017-01-05 15:14:40.331200  treatment     old_page          0
290360   714840  2017-01-10 23:35:22.559510  treatment     old_page          1
290647   904581  2017-01-17 11:35:54.031953  treatment     old_page          0
291313   807667  2017-01-15 19:11:59.976235  treatment     old_page          0
291754   795252  2017-01-19 02:43:07.343575  treatment     old_page          1
291922   634098  2017-01-07 23:45:07.976016  treatment     old_page          0
292412   693843  2017-01-09 06:31:48.749071  treatment     old_page          1
292521   689329  2017-01-06 03:58:15.546309  treatment     old_page          0
292607   699462  2017-01-17 23:54:08.826755  treatment     old_page          0
292800   712112  2017-01-14 23:33:41.083796  treatment     old_page          0
292963   742202  2017-01-12 04:34:20.344485  treatment     old_page          0
292977   638460  2017-01-22 13:38:30.677806  treatment     old_page          0
293240   861420  2017-01-04 20:34:09.065070  treatment     old_page          0
293302   825937  2017-01-04 20:56:48.825875  treatment     old_page          0
293391   934444  2017-01-12 19:49:35.581289  treatment     old_page          0
293443   738761  2017-01-04 15:20:52.694440  treatment     old_page          0
293530   934040  2017-01-04 20:52:26.981566  treatment     old_page          0
293773   688144  2017-01-16 20:34:50.450528  treatment     old_page          1
293817   876037  2017-01-17 16:15:08.957152  treatment     old_page          1
293917   738357  2017-01-05 15:37:55.729133  treatment     old_page          0
294014   813406  2017-01-09 06:25:33.223301  treatment     old_page          0
294252   892498  2017-01-22 01:11:10.463211  treatment     old_page          0

[1965 rows x 5 columns]
```

In [8]:
```
df.query('group=="control" and landing_page != "old_page"')
```

Out[8]:

	user_id	timestamp	group	landing_page	converted
22	767017	2017-01-12 22:58:14.991443	control	new_page	0
240	733976	2017-01-11 15:11:16.407599	control	new_page	0
490	808613	2017-01-10 21:44:01.292755	control	new_page	0
846	637639	2017-01-11 23:09:52.682329	control	new_page	1
850	793580	2017-01-08 03:25:33.723712	control	new_page	1
988	698120	2017-01-22 07:09:37.540970	control	new_page	0
1198	646342	2017-01-06 18:39:23.484797	control	new_page	0
1354	735021	2017-01-16 09:51:29.349493	control	new_page	0
1474	678638	2017-01-18 06:36:42.515395	control	new_page	0
1877	717682	2017-01-07 03:05:39.891873	control	new_page	0
2023	937692	2017-01-19 01:29:42.739007	control	new_page	0
2214	649781	2017-01-20 03:50:20.837704	control	new_page	0
2745	872666	2017-01-05 07:44:32.050781	control	new_page	0
2759	639817	2017-01-06 23:39:11.754971	control	new_page	0
2857	738999	2017-01-08 15:21:55.309961	control	new_page	0
2947	847673	2017-01-07 18:45:04.253063	control	new_page	1
3362	858458	2017-01-06 04:51:33.183576	control	new_page	1
3421	638068	2017-01-20 01:57:00.012096	control	new_page	0
3548	807355	2017-01-21 11:10:28.793058	control	new_page	0

```
3817     832098  2017-01-15 06:06:26.163307  control     new_page          0
3903     855630  2017-01-10 16:24:01.119709  control     new_page          1
3913     937090  2017-01-22 07:38:49.397402  control     new_page          0
4038     919582  2017-01-04 12:24:28.755065  control     new_page          0
4282     866677  2017-01-24 05:04:14.004157  control     new_page          0
4284     847508  2017-01-03 19:31:14.396402  control     new_page          0
4311     924330  2017-01-23 07:08:56.964247  control     new_page          0
4485     838568  2017-01-15 04:02:13.337797  control     new_page          0
4693     799659  2017-01-22 09:50:16.421384  control     new_page          0
4748     872738  2017-01-08 02:16:03.976589  control     new_page          0
4962     729859  2017-01-19 14:17:09.976523  control     new_page          0
...         ...                         ...      ...          ...        ...
290811   931254  2017-01-19 03:56:48.943007  control     new_page          0
291093   922957  2017-01-12 00:58:45.303371  control     new_page          0
291100   810979  2017-01-07 18:48:46.403714  control     new_page          0
291240   807517  2017-01-22 10:07:39.903169  control     new_page          0
291358   929094  2017-01-11 03:52:10.013362  control     new_page          0
291423   848305  2017-01-19 07:30:03.635089  control     new_page          0
291728   828985  2017-01-02 13:55:08.790046  control     new_page          0
291839   740434  2017-01-13 07:04:11.067609  control     new_page          0
291876   766031  2017-01-03 22:49:27.025028  control     new_page          0
291946   861129  2017-01-12 19:00:59.118294  control     new_page          1
292147   746367  2017-01-10 04:37:37.933511  control     new_page          0
292178   645830  2017-01-14 11:12:33.289733  control     new_page          0
292235   679326  2017-01-07 07:27:46.910711  control     new_page          0
292239   908003  2017-01-22 15:17:03.083738  control     new_page          0
292405   819974  2017-01-03 05:58:44.734645  control     new_page          0
292570   778969  2017-01-21 12:59:42.740399  control     new_page          1
292748   684361  2017-01-19 03:59:57.656614  control     new_page          0
292845   893018  2017-01-10 15:05:37.522921  control     new_page          0
293017   792268  2017-01-06 09:21:58.341063  control     new_page          0
293085   884635  2017-01-19 14:19:48.484389  control     new_page          0
293393   636565  2017-01-12 07:26:31.103374  control     new_page          0
293480   638376  2017-01-18 15:41:02.395882  control     new_page          0
293568   704024  2017-01-15 17:06:09.309987  control     new_page          0
293662   927109  2017-01-04 09:14:33.647192  control     new_page          0
293888   865405  2017-01-12 08:38:50.511434  control     new_page          0
293894   741581  2017-01-09 20:49:03.391764  control     new_page          0
293996   942612  2017-01-08 13:52:28.182648  control     new_page          0
294200   928506  2017-01-13 21:32:10.491309  control     new_page          0
294253   886135  2017-01-06 12:49:20.509403  control     new_page          0
294331   689637  2017-01-13 11:34:28.339532  control     new_page          0

[1928 rows x 5 columns]
```

f. Do any of the rows have missing values?

```
In [9]: #checking for null or missing values
        df.info()
```

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

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 [10]: df2_treatment = df.query('group == "treatment" and landing_page == "new_page"')

In [11]: df2_control = df.query('group == "control" and landing_page == "old_page"')

In [12]: df2 = df2_control.merge(df2_treatment, how ='outer')

In [13]: df2.shape

Out[13]: (290585, 5)

In [14]: df2.head()

Out[14]:    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   864975  2017-01-21 01:52:26.210827  control     old_page          1
         3   936923  2017-01-10 15:20:49.083499  control     old_page          0
         4   719014  2017-01-17 01:48:29.539573  control     old_page          0

In [15]: # Double Check all of the correct rows were removed - this should be 0
         df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].sh

Out[15]: 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 [16]: df2.nunique()
```

```
Out[16]: 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 [17]: sum(df2.user_id.duplicated())
```

```
Out[17]: 1
```

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

```
In [18]: df2[df2.duplicated(['user_id'], keep= False)]
```

```
Out[18]:         user_id                   timestamp      group landing_page  converted
         146212   773192  2017-01-09 05:37:58.781806  treatment     new_page          0
         146678   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 [19]: df2= df.drop_duplicates(subset = "user_id", keep = 'first')
```

```
In [20]: df2.shape
```

```
Out[20]: (290584, 5)
```

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

```
Out[21]: 0.11956955647936569
```

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

```
In [22]: df2_control = df2.query('group == "control"')
         df2_control.converted.mean()
```

```
Out[22]: 0.12029717968491792
```

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

```
In [23]: df2_treatment = df2.query('group == "treatment"')
         df2_treatment.converted.mean()
```

```
Out[23]: 0.11884253398646046
```

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

```
In [24]: len(df2_treatment.index)
         len(df2.index)
         r_newpage= len(df2_treatment.index)/len(df2.index)
         r_newpage

Out[24]: 0.5002064807422294
```

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.

ANSWER: We can see from the above result that individuals in the treatment group had a conversion rate of 11.88% and individuals in the control grounp had a conversion rate of 12.04%.

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.

ANSWER: If we assume the old page is better unless the new page proves to be definately better, then the null hypotheses is that the mean converted rate of the old page is greater or equal to the converted rate of the new page and the alternative hypothesis is that the mean converted rate of the old page is less than the converted rate of the new page.

Null Hypotheses: p_{old} is equal greater or equal to p_{new}

Alternative Hypothesis: p_{old} is less than p_{new}

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 [25]:  = df2.converted.mean()
```

```
Out[25]: 0.11956955647936569
```

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

```
In [26]:  = df2.converted.mean()
```

```
Out[26]: 0.11956955647936569
```

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

```
In [27]: n_new = len(df2_treatment.index)
         n_new
```

```
Out[27]: 145352
```

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

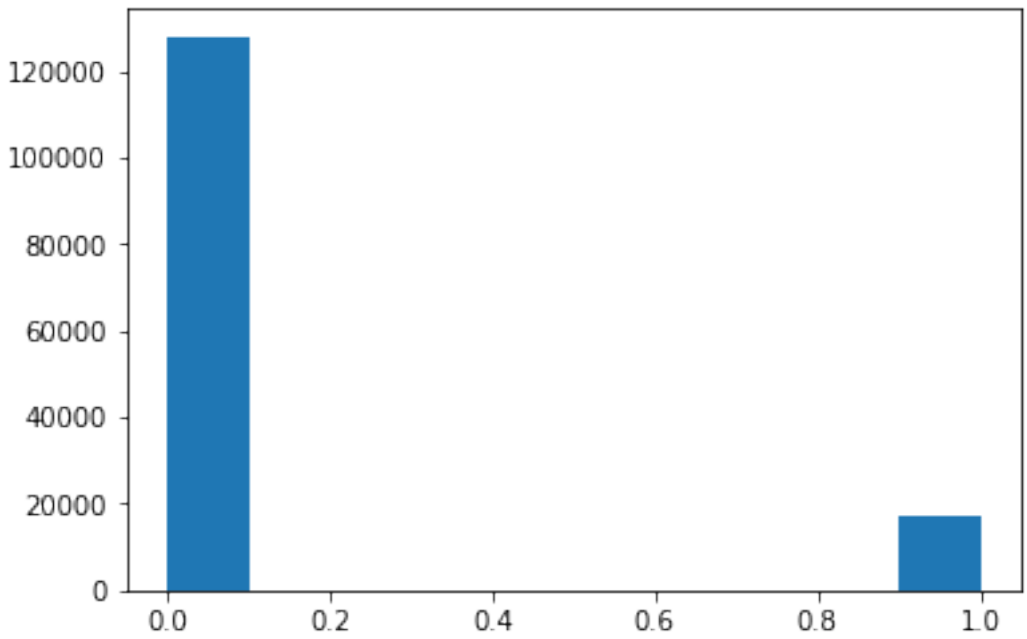
```
In [28]: n_old = len(df2_control.index)
         n_old
```

```
Out[28]: 145232
```

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**.

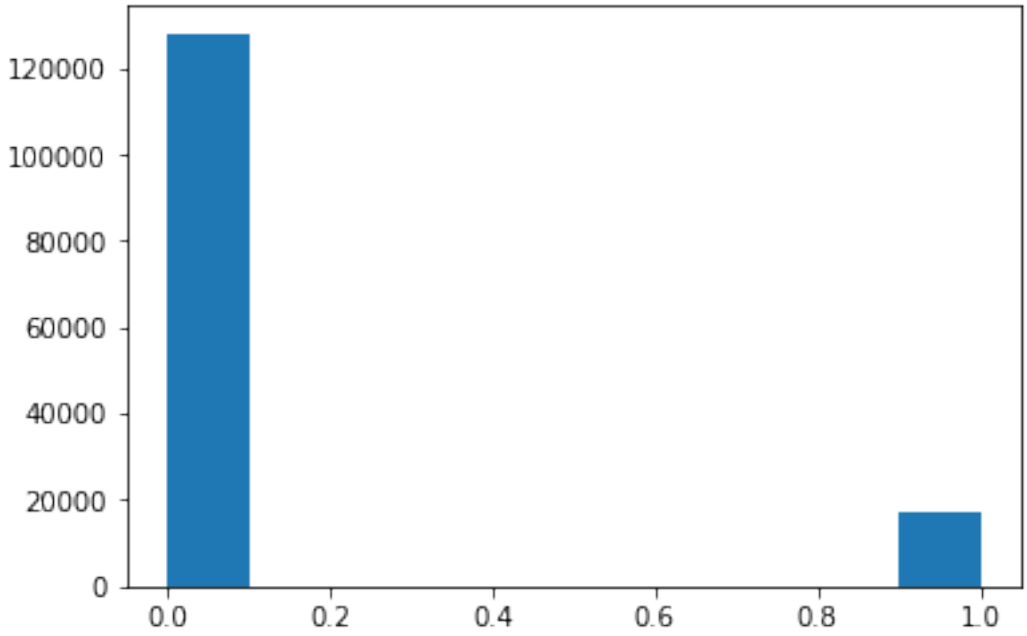
```
In [29]: new_page_converted = np.random.choice([1, 0], size=len(df2_treatment.index), p=[df2.con
```

```
In [30]: plt.hist(new_page_converted);
```



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 [31]: old_page_converted = np.random.choice([1, 0], size=len(df2_control.index), p=[df2.conve
         plt.hist(old_page_converted);
```



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

```
In [32]: new_page_converted.mean() - old_page_converted.mean()
```

```
Out[32]: 0.0018153125842889639
```

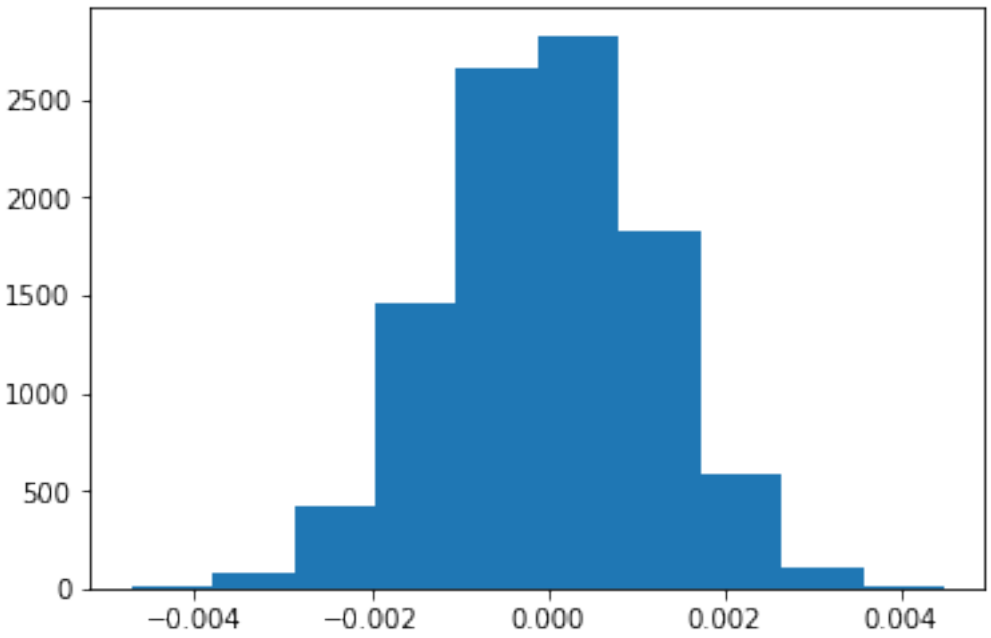
h. Create 10,000 p_{new} - p_{old} values using the same simulation process you used in parts (a) through (g) above. Store all 10,000 values in a NumPy array called **p_diffs**.

```
In [33]: p_diffs=[]

         for i in range (10000):
             new_page_converted = np.random.binomial(1, , n_new)
             old_page_converted = np.random.binomial(1, , n_old)
             new_page_p = new_page_converted.mean()
             old_page_p = old_page_converted.mean()
             p_diffs.append(new_page_p - old_page_p)
```

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 [34]: #showing the histogram
         plt.hist(p_diffs);
```



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

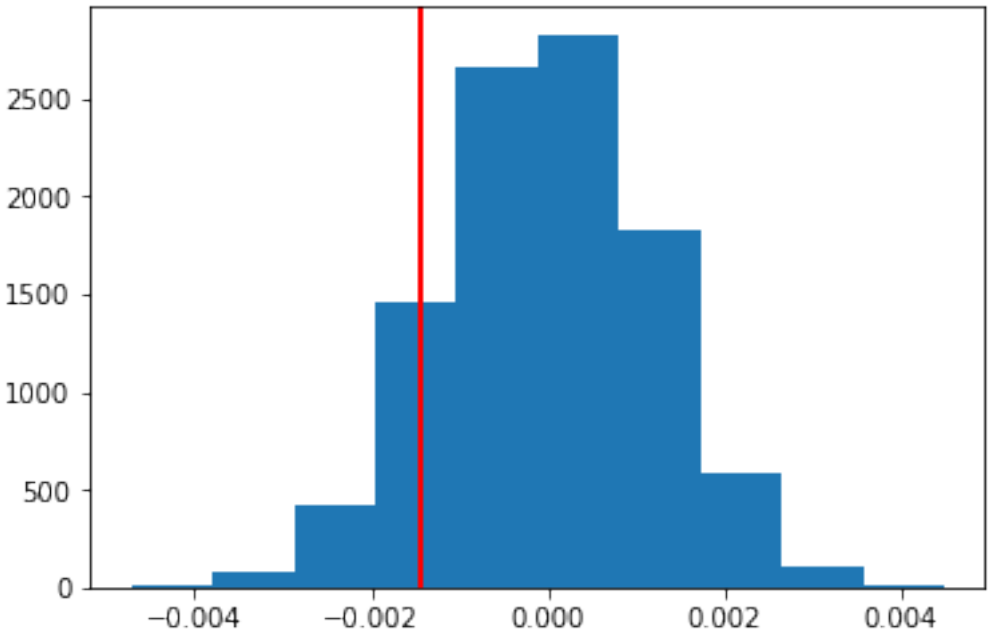
```
In [35]: #Actual difference of converted rates
         actual_diff = (df2[df2['group'] == "treatment"]['converted'].mean()) - (df2[df2['group'
         actual_diff

Out[35]: -0.0015456456984574629

In [36]: #Convert to numpy array and calculate the p-value
         p_diffs = np.array(p_diffs)
         (p_diffs > actual_diff).mean()

Out[36]: 0.88529999999999998

In [37]: plt.hist(p_diffs);
         plt.axvline(actual_diff,c='r',linewidth = 2);
```



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?

ANSWER: The percentage of 88.28 is called scientifically p-value, which determines the probability of obtaining our observed statistic (or one more extreme in favor of the alternative) if the null hypothesis is true.

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 [38]: import statsmodels.api as sm

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

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

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

         z_score, p_value

Out[39]: (1.2083846739740718, 0.88655033391933613)
```

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.**?

ANSWER: The positive z value is indicating that the score is above the mean and the p value is greater than 0.05; that means we don't have enough evidence to reject the null hypothesis.

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?

ANSWER: 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 [40]: #creating intercepting column
         df2['intercept'] = 1
         #creating dummies
         ab_page = ['treatment', 'control']
         df2['ab_page'] = pd.get_dummies(df2.group)['treatment']
         df2.head()

/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:2: 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#

/opt/conda/lib/python3.6/site-packages/ipykernel_launcher.py:5: 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#
  """
```

```
Out[40]:    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 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 [41]: logi = 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 [42]: results = logi.fit()
         results.summary2()

Optimization terminated successfully.
         Current function value: 0.366064
         Iterations 6


Out[42]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                  Results: Logit
         =================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2020-05-21 16:50 AIC:              212748.6664
         No. Observations:   290584           BIC:              212769.8257
         Df Model:           1                Log-Likelihood:   -1.0637e+05
         Df Residuals:       290582           LL-Null:          -1.0637e+05
         Converged:          1.0000           Scale:            1.0000
         -----------------------------------------------------------------
                       Coef.   Std.Err.      z      P>|z|    [0.025   0.975]
         -----------------------------------------------------------------
         intercept    -1.9896    0.0081  -246.6589  0.0000  -2.0054  -1.9738
         ab_page      -0.0138    0.0114    -1.2084  0.2269  -0.0362   0.0086
         =================================================================

         """
```

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**?

ANSWER: The p-value associated with the ab_page is 0.22, which is lower than the p-value found in Part(II); 0.88. * In Part II, A/B Test method we had: null hypothesis: <= , Alternative hypothesis p_new > p_old * In Part III under logistic model we have: null hypothesis: = , Alternative hypothesis p_new p_old and in this part the intercept has been added.

f. Now, you are considering other things that might influence whether or not an individual converts. Discuss why it is a good idea to consider other factors to add into your regression model. Are there any disadvantages to adding additional terms into your regression model?

ANSWER: we can consider "timestamp" becuase the conversion rate is likely to have some kind of relation with different dates and time like evenings when people get off work.
Disadvantage: the more terms , the model will be more complex

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 [43]: countries = pd.read_csv('countries.csv')
         #countries.head()
         #countries.country.unique()
         df_new= countries.set_index('user_id').join(df2.set_index('user_id'), how= 'inner')
         #df_new.head()
         df_new[['CA','UK','US']]=pd.get_dummies(df_new['country'])
         # df_new.head()
         df_new = df_new.drop('CA', axis=1)
         df_new.head()
```

Out[43]:

user_id	country	timestamp	group	landing_page	converted	intercept	ab_page	UK	US
834778	UK	2017-01-14 23:08:43.304998	control	old_page	0	1	0	1	0
928468	US	2017-01-23 14:44:16.387854	treatment	new_page	0	1	1	0	1
822059	UK	2017-01-16 14:04:14.719771	treatment	new_page	1	1	1	1	0
711597	UK	2017-01-22 03:14:24.763511	control	old_page	0	1	0	1	0
710616	UK	2017-01-16 13:14:44.000513	treatment	new_page	0	1	1	1	0

```
In [44]: df_new['intercept'] = 1
         logi = sm.Logit(df_new['converted'], df_new[['intercept','UK','US','ab_page']])
         results = logi.fit()
         results.summary2()

Optimization terminated successfully.
         Current function value: 0.366059
         Iterations 6


Out[44]: <class 'statsmodels.iolib.summary2.Summary'>
         """
                                  Results: Logit
         =================================================================
         Model:              Logit            No. Iterations:   6.0000
         Dependent Variable: converted        Pseudo R-squared: 0.000
         Date:               2020-05-21 16:50 AIC:              212749.9927
         No. Observations:   290584           BIC:              212792.3113
         Df Model:           3                Log-Likelihood:   -1.0637e+05
         Df Residuals:       290580           LL-Null:          -1.0637e+05
         Converged:          1.0000           Scale:            1.0000
         -----------------------------------------------------------------
                        Coef.   Std.Err.     z      P>|z|    [0.025   0.975]
         -----------------------------------------------------------------
         intercept     -2.0266    0.0266  -76.2252  0.0000  -2.0787  -1.9744
         UK             0.0459    0.0284    1.6200  0.1052  -0.0096   0.1015
         US             0.0363    0.0268    1.3516  0.1765  -0.0163   0.0889
         ab_page       -0.0138    0.0114   -1.2043  0.2285  -0.0362   0.0086
         =================================================================

         """

In [45]: print(np.exp(results.params))
         print('\n')
         #for values less than 1, reciprocal it for better explanation
         print(1/np.exp(results.params))

intercept    0.131789
UK           1.047010
US           1.036950
ab_page      0.986322
dtype: float64


intercept    7.587913
UK           0.955101
US           0.964367
ab_page      1.013867
```

```
dtype: float64
```

- for ab_page: p-value is 0.22
- for UK: conversion is 1.04 times or 4.7% more likely to occur while holding everything as constant
- for US: conversion is 1.03 times or 3.6% more likely to occur while holding everything as constant

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 [46]: df_new['UK_page']=df_new.ab_page*df_new.UK
         df_new['US_page']=df_new.ab_page*df_new.US
         df_new.head()
```

Out[46]:

user_id	country	timestamp	group	landing_page	converted	intercept	ab_page	UK	US	UK_page	US_page
834778	UK	2017-01-14 23:08:43.304998	control	old_page	0	1	0	1	0	0	0
928468	US	2017-01-23 14:44:16.387854	treatment	new_page	0	1	1	0	1	0	1
822059	UK	2017-01-16 14:04:14.719771	treatment	new_page	1	1	1	1	0	1	0
711597	UK	2017-01-22 03:14:24.763511	control	old_page	0	1	0	1	0	0	0
710616	UK	2017-01-16 13:14:44.000513	treatment	new_page	0	1	1	1	0	1	0

```
In [47]: logi=sm.Logit(df_new['converted'],df_new[['intercept','ab_page','UK','US','UK_page','US
         results=logi.fit()
         results.summary2()
```

```
Optimization terminated successfully.
         Current function value: 0.366056
         Iterations 6
```

Out[47]:

```
<class 'statsmodels.iolib.summary2.Summary'>
"""
                          Results: Logit
=================================================================
Model:              Logit            No. Iterations:   6.0000
```

```
Dependent Variable: converted        Pseudo R-squared: 0.000
Date:               2020-05-21 16:50 AIC:              212752.1135
No. Observations:   290584           BIC:              212815.5914
Df Model:           5                Log-Likelihood:   -1.0637e+05
Df Residuals:       290578           LL-Null:          -1.0637e+05
Converged:          1.0000           Scale:            1.0000
------------------------------------------------------------------
               Coef.   Std.Err.     z      P>|z|    [0.025   0.975]
------------------------------------------------------------------
intercept     -2.0026    0.0364  -54.9641  0.0000  -2.0740  -1.9312
ab_page       -0.0620    0.0519   -1.1942  0.2324  -0.1638   0.0398
UK             0.0111    0.0398    0.2794  0.7799  -0.0670   0.0892
US             0.0145    0.0377    0.3856  0.6998  -0.0593   0.0884
UK_page        0.0700    0.0567    1.2341  0.2172  -0.0412   0.1812
US_page        0.0438    0.0537    0.8153  0.4149  -0.0615   0.1490
==================================================================

"""
```

In [48]:
```
print(np.exp(results.params))
print('\n')
#for values less than 1, reciprocal it for better explanation
print(1/np.exp(results.params))
```

```
intercept    0.134986
ab_page      0.939870
UK           1.011197
US           1.014639
UK_page      1.072511
US_page      1.044753
dtype: float64


intercept    7.408187
ab_page      1.063977
UK           0.988927
US           0.985573
UK_page      0.932391
US_page      0.957164
dtype: float64
```

- Except for the intercept, the rest of the coefficients have a p value greater than 0.05 indicating the values are not statistically significant.
- UK: Conversion is 1.011 times more likely to occur if the user is from UK while holding everything else constant.
- US: Conversion is 1.014 times more likely to occur if the user is from US while holding everything else constant.

- UK Page: Conversion is 1.07 times more likely to occur for interaction between UK and page while holding everything else constant.
- US Page: Conversion is 1.04 times more likely to occur for interaction between US and page while holding everything else constant.

Finishing Up

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

```
Out[49]: 0
```

```
In [ ]:
```