FRESH Analyze_ab_test_results_notebook

June 1, 2020

0.1 Analyze A/B Test Results

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.

[1]:
```
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
%matplotlib inline
#and, though this was NEVER indicated in the lessons or videos, it will be␣
↪necessary for the logistic regression
from scipy import stats
stats.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df)
#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:

[2]:
```
df1 = pd.read_csv('ab_data.csv')
df1.head(5)
```

[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

There are five columns. The initial assumption is that the "landing_page" column and/or "group" column are the x-variables with the "converted" column being the y-variable against which x will be measured.

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

[3]:
```
df1.shape
```

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

There are 294,478 rows in the dataset.

c. The number of unique users in the dataset.

[4]:
```
df1.user_id.nunique()
```

[4]:
```
290584
```

There are 290,584 unique users in the dataset.

[5]:
```
df1.nunique()
```

[5]:
```
user_id         290584
timestamp       294478
group                2
landing_page         2
converted            2
dtype: int64
```

[6]:
```
df1.dtypes
```

[6]:
```
user_id          int64
timestamp       object
group           object
landing_page    object
```

```
converted        int64
dtype: object
```

Optional: This analyst is choosing to see how many unique responses there are in each column as well as the types of those responses. This may prove useful later in the analysis when determining how best to query.

d. The proportion of users converted.

[7]:
```python
df1['converted'].value_counts()
```

[7]:
```
0    259241
1     35237
Name: converted, dtype: int64
```

[8]:
```python
print (len(df1['converted']))
```

```
294478
```

The proportion of users converted is equal to the number of those converted divided by the number of all those converted. This should come out to:

35,237 / 294,478 = .119659

[9]:
```python
df1.converted.mean()
```

[9]:
```
0.11965919355605512
```

Optional: This is simpler way of attaining the same data and serves as a second validating check on the converted rate.

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

[10]:
```python
len(df1[(df1['landing_page']=='new_page') == (df1['group']!='treatment')])
```

[10]:
```
3893
```

The "new_page" and "treatment" pages don't line up 3,893 times.

f. Do any of the rows have missing values?

[11]:
```python
df1.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
```

All of the columns show the same number of rows with responses (294,478). So, none of the rows have missing values.

2. For the rows where **treatment** is not aligned with **new_page** or **control** is not aligned with **old_page**, we cannot be sure if this row truly received the new or old page. Use **Quiz 2** in the classroom to provide how we should handle these rows.

a. Now use the answer to the quiz to create a new dataset that meets the specifications from the quiz. Store your new dataframe in **df2**.

[12]:
```
df2 = df1
```

Dataframe 2 (df2) will now be an offshoot of dataframe 1 (df1) allowing us to work this section without altering the data analysis already performed in dataframe 1.

[13]:
```
# Double Check all of the correct rows were removed - this should be 0
df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) ==␣
↪False].shape[0]
```

[13]:
```
3893
```

[14]:
```
# Google search for delete rows based on conditional clause from seperate␣
↪columns
# delete all rows for which column 'Age' has value greater than 30 and Country␣
↪is India
# indexNames = dfObj[ (dfObj['Age'] >= 30) & (dfObj['Country'] == 'India') ].
↪index
# dfObj.drop(indexNames , inplace=True)
```

[15]:
```
#rmv_rows = df2[ (df2['group'] == 'treatment') & (df2['landing_page']␣
↪=='new_page') ].index
#df2.drop(rmv_rows , inplace=True)

rmv_rows = df1[((df1['group'] == 'treatment') == (df1['landing_page']␣
↪=='new_page')) == False].index
df2.drop(rmv_rows , inplace=True)
```

[16]:
```
# Double Check all of the correct rows were removed - this should be 0
df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) ==␣
↪False].shape[0]
```

[16]:
```
0
```

[17]:
```
df2.shape
```

[17]:
```
(290585, 5)
```

df1 has 294,478 rows and, now df2 has 290,585. This means that the 3,893 rows that were observed as being out of alignement (landing_page column vs. group column) were just deleted. Now, the number of rows that hold values in the landing_page column is congruent with the number of rows that hold values in the group column.

[18]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 290585 entries, 0 to 294477
Data columns (total 5 columns):
user_id          290585 non-null int64
timestamp        290585 non-null object
group            290585 non-null object
landing_page     290585 non-null object
converted        290585 non-null int64
dtypes: int64(2), object(3)
memory usage: 13.3+ MB
```

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**?

[19]:
```
df2.nunique()
```

[19]:
```
user_id         290584
timestamp       290585
group                2
landing_page         2
converted            2
dtype: int64
```

There are 290,584 unique user_id's in df2.

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

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

[20]:
```
sum(df2.duplicated('user_id'))
```

[20]:
```
1
```

[21]:
```
duprow = df2[df2.duplicated(['user_id'])]
print("Duplicated rows based on user_id column:", duprow, sep='\n')
```

```
Duplicated rows based on user_id column:
      user_id                   timestamp      group landing_page  converted
2893   773192  2017-01-14 02:55:59.590927  treatment     new_page          0
```

The one user_id repeated in df2 is 773192. The landing_page for that user_id is new_page, the group is treatment and the converted value is 0 (not converted).

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

```
[22]: #drop duplicated row
df2.drop_duplicates(subset='user_id', keep="last", inplace=True)
```

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

```
[23]: 0
```

These commands will drop in place while still maintaining all data within DF2.

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

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

```
[24]: df2.head(10)
```

```
[24]:    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
      5   936923  2017-01-10 15:20:49.083499    control     old_page          0
      6   679687  2017-01-19 03:26:46.940749  treatment     new_page          1
      7   719014  2017-01-17 01:48:29.539573    control     old_page          0
      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
```

```
[25]: df2.shape
```

```
[25]: (290584, 5)
```

```
[26]: df2.converted.sum()
```

```
[26]: 34753
```

The probility of an individual converting regardless of the page they receive is equal to the sum of the converted rows (numerator) divided by the number of unique users, or the shape of all of df2 (denominator) or:

34753 / 290584 = .119597

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

```
[27]: 0.11959708724499628
```

Optional: the rate can be computed by .mean( ) as well.

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

```
[28]: df2.groupby('group')['converted'].mean()
```

[28]:
```
group
control      0.120386
treatment    0.118808
Name: converted, dtype: float64
```

Given that an individual was in the control group, the probability that they converted was .1204

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

[29]:
```
df2.groupby('group')['converted'].mean()
```

[29]:
```
group
control      0.120386
treatment    0.118808
Name: converted, dtype: float64
```

Given that an individual was in the treatment group, the probability that they converted was .1188

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

[30]:
```
print (len(df2[df2['landing_page']=='new_page']))
print (df2.shape[0])
```

```
145310
290584
```

[31]:
```
print (len(df2[df2['group']=='treatment']))
print (df2.shape[0])
```

```
145310
290584
```

Optional: This is just for a check to make sure group and landing_page match up from the previous work completed (they should).

[32]:
```
(len(df2[df2['landing_page']=='new_page']))/df2.shape[0]
```

[32]:
```
0.5000619442226688
```

The probability that an individual receives a new page is 0.50006. The implication is that a) the probability that an individual receives an old page is also 0.50, and b) that the probability of receiving either page is split 50/50.

[33]:
```
(len(df2[df2['group']=='treatment']))/df2.shape[0]
```

[33]: `0.5000619442226688`

Optional: Again, verification that landing_page and group page show the same probability (they should).

e. Consider your results from a. through d. above, and explain below whether you think there is sufficient evidence to say that the new treatment page leads to more conversions.

Has anything done thus far indicate that the new website converts more customers to purchasing products than the old website? It's this analyst's opinion that there hasn't been descriptive statistical evidence to indicate that website visitors receving the new web page are more likely to purchase and convert than those visitors who received the old website page. In fact, just observing the proportions, the old page holds a slight lead over the new page in terms of the proportion of conversions.

The recommendation would be for a deeper analysis via A/B testing where the current data can be bootstrapped and analyzed further. However, if that analysis weren't approved by management, then this analyst would not recommend funding a new page solely in hopes of increasing conversions because the data (as it is currently described to this point) doesn't support that action.

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.

Assuming that the old page is better than or equal to the new page (as stated above in the customer requirements) then the null and alternative hypothesis are as follows:

H0 : Pold Conversion Rate Pnew Conversion Rate

H1 : Pold Conversion Rate < Pnew Conversion Rate

`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.

Now, assuming that the old page is equal to the new page (as stated in these particular customer requirements), then the null and alternative hypothesis look ike this:

> H0 : Pold Conversion Rate = Pnew Conversion Rate (both equaling .119597, the df2.converted.mean)
> H1 : Pold Conversion Rate Pnew Conversion Rate

INCREDIBLY IMPORTANT NOTE: it's a valuable reminder here that per these particular customer requirements, the construction of the hypothesis test is being set up with the H0 stating that the "old pager's" and "new pagers's" converted rates are EQUAL TO EACH OTHER and the H1 with them NOT BEING EQUAL to each other. Why is this important to note while testing? Because it sets in motion that all levels of significance will be split evenly in a two-tailed test.

Based on my personal experience running hypothesis testing in my previous career, the majority of the customer requirements choose to show the alternative hypothesis as greater than (or less than) X1 which forces H0 to be a less than or equal to (or greater than or equal) statement. For whatever reason, that wasn't the above requirements for this udacity assignment.

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

[34]:
```python
p_new = df2.converted.mean()
p_new
```

[34]:
```
0.11959708724499628
```

The conversion rates for Pold and pnew are the same (per the requirements). That rate was previously calulated using .mean( ) and is 0.11597.

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

[35]:
```python
p_old = df2.converted.mean()
p_old
```

[35]:
```
0.11959708724499628
```

The conversion rates for Pold and pnew are the same (per the requirements). That rate was previously calulated using .mean( ) and is 0.11597.

c. What is n_{new}?

[36]:
```python
df2.landing_page.value_counts()
```

[36]:
```
new_page    145310
old_page    145274
Name: landing_page, dtype: int64
```

[37]:
```python
n_new_page = (len(df2[df2['landing_page']=='new_page']))
print (n_new_page)
```

```
145310
```

Using a sample size for each page equal to the ones in ab_data.csv (df2), nnew is 145,310. So, 145,310 site visitors received the new page.

d. What is n_{old}?

[38]:
```python
df2.landing_page.value_counts()
```

[38]:
```
new_page    145310
old_page    145274
Name: landing_page, dtype: int64
```

[39]:
```python
n_old_page = (len(df2[df2['landing_page']=='old_page']))
print (n_old_page)
```

```
145274
```

Using a sample size for each page equal to the ones in ab_data.csv (df2), nold is 145,274. So, 145,274 site visitors received the old page.

e. Simulate n_{new} transactions with a convert rate of p_{new} under the null. Store these n_{new} 1's and 0's in **new_page_converted**.

> Just a reminder, per the customer requirements above, this is the Null and Alternative hypotheses:
>
> H0 : Pold Conversion Rate = Pnew Conversion Rate
>
> H1 : Pold Conversion Rate Pnew Conversion Rate

Simulating the convert rate of Pnew & old using nnew & old (from the samples of new_page & old_page), a single binomial experiement is run to find the convert rate of the "new pagers" & the convert rate of the "old pagers".

[40]:
```python
# example from binomial dist, Lessons 11.4
new_page_converted = np.random.binomial(1,p_new,n_new_page)
# here's the mean of that too
(new_page_converted==1).mean()
```

[40]:
```
0.11955818594728511
```

f. Simulate n_{old} transactions with a convert rate of p_{old} under the null. Store these n_{old} 1's and 0's in **old_page_converted**.

Again, simulating the convert rate of Pnew & old using nnew & old (from the samples of new_page & old_page), a single binomial experiement is run to find the convert rate of the "new pagers" & the convert rate of the "old pagers".

```
[41]: # example from binomial dist, Lessons 11.4
old_page_converted = np.random.binomial(1,p_old,n_old_page)
# here's the mean of that too
(old_page_converted==1).mean()
```

```
[41]: 0.1183763095942839
```

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

This shows the differnce of the convert rates between "old pagers" and "new pagers" after running a single binomial experiment and comparing the difference in their resultant means.

```
[42]: one_dif_mean = new_page_converted.mean() - old_page_converted.mean()
one_dif_mean
```

```
[42]: 0.00118187635300121
```

There is 1/1000th's of a differnce in their proportions of means converted. That's not a lot of difference between the converted mean of the new page and the converted mean of the old page!

However, that's only one observation of a binomial experiment...

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

Bootstrap sampling the exact same experiment 10,000 times (mean of new page converts, mean of old page converts and the difference of their two means) allows for the creation of a sampling distribution of those 10,000 pieces of data and, more importantly, binds those three sampling distributions to all the statistical "rules" that a normally distributed sampling must adhere to(eg., measures of central tendency, etc.)

```
[43]: new_pg_cv_means, old_pg_cv_means, p_diffs = [], [], []
for _ in range(10000):
    new_page_converted = np.random.binomial(1,p_new,n_new_page)
    new_page_converted.mean()
    old_page_converted = np.random.binomial(1,p_old,n_old_page)
    old_page_converted.mean()
    new_pg_cv_means.append(new_page_converted.mean())
    old_pg_cv_means.append(old_page_converted.mean())
    p_diffs.append(new_page_converted.mean() - old_page_converted.mean())
np.percentile(p_diffs, 2.5), np.percentile(p_diffs, 97.5)
```

```
[43]: (-0.002348958121558306, 0.002406865834101514)
```

Optional:

Though including the Confidence Interval isn't requested, nor required, for Hypothesis Testing and simulating from the null, it is a valid precursor to let an analyst know which way simulating from the null will lean.

In this case, 0 is included in our interval which means that having a zero as an outcome is well within the realm of possibilty at this level of signicancce. It further implies that there won't be evidence to support rejecting the null hypothesis. Or, in words, this evidence doesn't support rejecting the idea that the mean convert rate of "old pagers" equals the "new pagers".

As we proceed, we'll look to verify this through the remainder of the Hypothesis test.

[44]:
```
new_pg_cv_means, old_pg_cv_means, p_diffs = [], [], []
for _ in range(10000):
    new_page_converted = np.random.binomial(1,p_new,n_new_page)
    new_page_converted.mean()
    old_page_converted = np.random.binomial(1,p_old,n_old_page)
    old_page_converted.mean()
    new_pg_cv_means.append(new_page_converted.mean())
    old_pg_cv_means.append(old_page_converted.mean())
    p_diffs.append(new_page_converted.mean() - old_page_converted.mean())
```

[45]:
```
#std dev of the sampling dist of new page converted rates
np.std(new_pg_cv_means)
```

[45]:
```
0.0008444563570129727
```

[46]:
```
#std dev of the sampling dist of old page converted rates
np.std(old_pg_cv_means)
```

[46]:
```
0.0008532178154928046
```

[47]:
```
#std dev of the sampling dist of the differences of the converted rates
np.std(p_diffs)
```

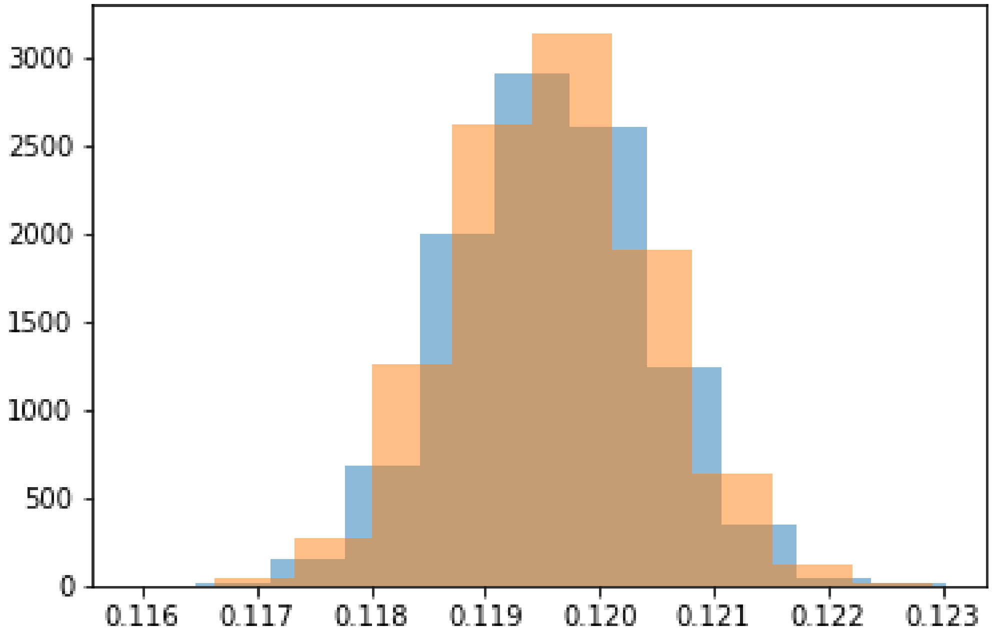
[47]:
```
0.001197887254528692
```

Same as the results of the single binomial experiment, when sampled 10,000 time there is still 1/1000th's of a differnce in the proportions of means converted between "old pagers" and "new pagers".

The histogram shows the close proximity of the bootstrapped converted rates of "old pagers" and "new pagers".

[48]:
```
plt.hist(new_pg_cv_means, alpha = 0.5)
plt.hist(old_pg_cv_means, alpha = 0.5)
```

[48]:
```
(array([3.000e+00, 3.600e+01, 2.690e+02, 1.264e+03, 2.625e+03, 3.143e+03,
        1.903e+03, 6.280e+02, 1.150e+02, 1.400e+01]),
 array([0.11592577, 0.11662376, 0.11732175, 0.11801974, 0.11871773,
        0.11941572, 0.12011372, 0.12081171, 0.1215097 , 0.12220769,
        0.12290568]),
 <a list of 10 Patch objects>)
```



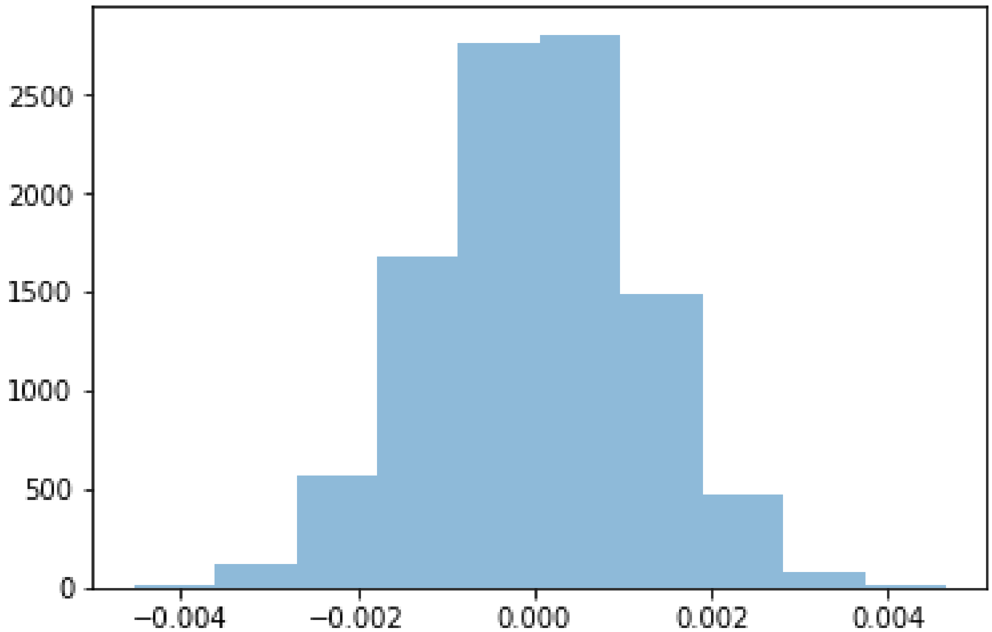
i. Plot a histogram of the **p_diffs**. Does this plot look like what you expected? Use the matching problem in the classroom to assure you fully understand what was computed here.

[49]:
```
act_ctrl_cv = df2.query('group == "control"')['converted'].mean()
act_trtmt_cv = df2.query('group == "treatment"')['converted'].mean()
obs_diff = act_trtmt_cv - act_ctrl_cv
print("Multiple sources as well as StackOverFlow state that I must code in this␣
↪fashion.  Here is the observed statistic: {}".format(obs_diff))
```

```
Multiple sources as well as StackOverFlow state that I must code in this
fashion.  Here is the observed statistic: -0.0015782389853555567
```

[50]:
```
plt.hist(p_diffs, alpha = 0.5)
```

[50]:
```
(array([  14.,  116.,  565., 1677., 2768., 2806., 1494.,  474.,   75.,
          11.]),
 array([-4.53783556e-03, -3.61896410e-03, -2.70009263e-03, -1.78122117e-03,
        -8.62349706e-04,  5.65217568e-05,  9.75393220e-04,  1.89426468e-03,
         2.81313615e-03,  3.73200761e-03,  4.65087907e-03]),
 <a list of 10 Patch objects>)
```

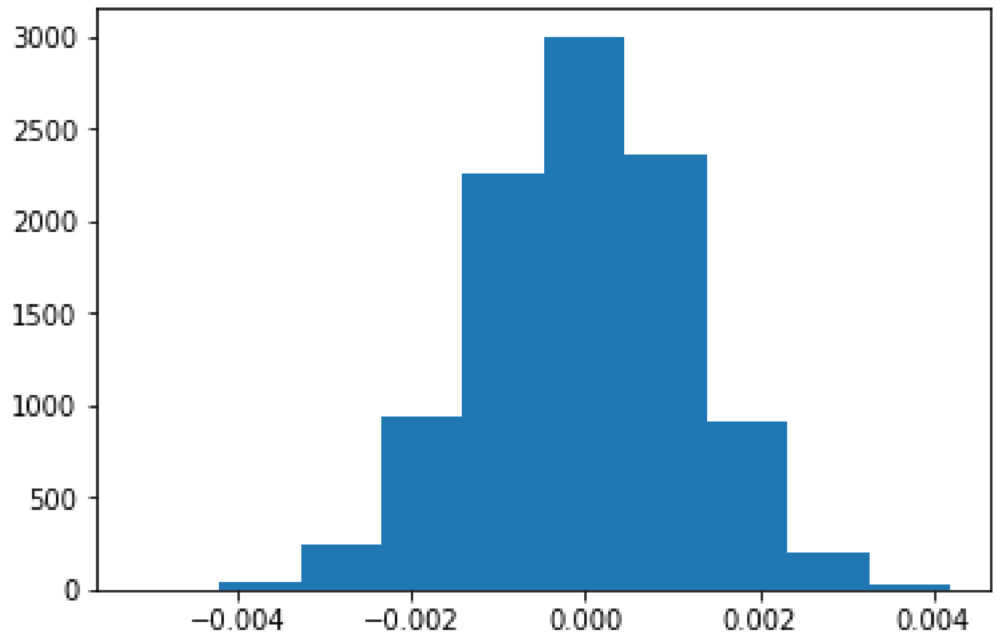


The differences in means histogram from the bootstrapped sample further supports that the range of potential differences between the "oldpagers" and "new pagers" conversion is very close. In general, you could expect to find anywhere from -0.004 to 0.004 difference in the converted rates and still consider that a "normal" answer (meaning not abnormal or significant).

Now, we'll use the sampling distribution for the difference in means to simulate what you'd expect if the sampling distribution were centered on zero.

[51]:
```
#10000 draws from the sampling distribution under the null
null_vals = np.random.normal(0, np.std(p_diffs), 10000)
```

[52]:
```
plt.hist(null_vals);
```



[53]:
```
#this should be incredibly close to zero
np.mean(null_vals)
```

[53]: `-2.1819103670217107e-05`

The mean of the 10,000 draws from the null_vals df is incredibly close to zero (-0.000021819103670217107). Meaning, there is no difference in the means of the "old pagers" converted rate and the "new pagers" because the mean of the null_vals between them is essentially zero.

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

[54]:
```
#There was a tremendous level of searching out for this answer
#convert to numpy array so there isn't a "list and float" below.
p_diffs = np.array(p_diffs)
```

[55]:
```
# https://knowledge.udacity.com/questions/26449
print("Here is the proportion of the p_diffs that is greater than the actual␣
↪difference observed: {}".format((p_diffs > obs_diff).mean()))
```

```
Here is the proportion of the p_diffs that is greater than the actual difference
observed: 0.9063
```

k. In words, explain what you just computed in part **j.** What is this value called in scientific studies? What does this value mean in terms of whether or not there is a difference between the new and old pages?

The answer to 2g above showed very little difference between the converted rate of the "new pagers" and the "old pagers" (approx 0.0018585569337940672 difference) making them essentially equal to each other and H0 was confirmed (albeit, very informally since it was only one sample.)

Optional: The answer to 2h above showed through the Confidence Interval (though neither requested, nor required, for Hypothesis Testing and simulating from the null), that 0 was included in the interval, suggesting that there wouldn't be evidence to support a statistically significant difference in the converted means of the new page & old page.

~~**The answer to 2j goes further and confirms that there is very, very little difference between the initial assumption of an insignificant difference between "old pager" and "new pagers" converted rate via one sample and the converted rate when bootstrapped and run 10,000 times (there is zero difference between the two (actually, -0.000005678744028457579 difference and not 0.0)).**~~

Amended answer to 2j: The analyst initially misunderstood the question being asked of him. What's being calculatd here is the probablity of the observed statistic falling within the sampling distribution. Because this p-vaule is large it still supports null hypothesis and supports the previous findings. This value should be approximately equal to the value in 2m

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.

   One must assume that this question is requring the number of converts of the "old pagers" and the "new pagers" as well as the overall number of "old pagers" and "new pagers". All of these must be coming out of df2 which is the adjusted data frame with the duplicated rows already removed.

[56]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 290584 entries, 0 to 294477
Data columns (total 5 columns):
user_id         290584 non-null int64
timestamp       290584 non-null object
group           290584 non-null object
landing_page    290584 non-null object
converted       290584 non-null int64
dtypes: int64(2), object(3)
memory usage: 13.3+ MB
```

[57]:
```
import statsmodels.api as sm

convert_old = df2.query('group == "control"')['converted'].sum()
convert_new = df2.query('group == "treatment"')['converted'].sum()
n_old = df2.query('landing_page == "old_page"').shape[0]
```

```
n_new = df2.query('landing_page == "new_page"').shape[0]
```

Running a 2 proportion hypothesis test using the P-Value approach requires four values at the onset: x1, x2, n1 & n2.

x 1 & 2 represent the number of occurences that meet a specific trait or condition.

n 1 & 2 represent the number of the sample upon which the occurences reside.

In the above example, convert_old & convert_new represent x 1 & 2 (respectively) and n_old & n_new represent n 1 & 2 (respectively).

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

   INCREDIBLY IMPORTANT NOTE (revisited): it's a valuable reminder here that per the customer requirements, the hypothesis test was set up with the null hypothesis setting "old pagers" and "new pagers" converted rates EQUAL TO EACH OTHER and the alternative hypothesis with them being NOT BEING EQUAL to each other. Why is this important to note while testing? Because it sets in motion that all levels of significance will be split evenly in a two-tailed test rather than a single-sided test.

   Based on my personal experience running hypothesis testing in my previous career, the majority of the customer requirements choose to show the alternative hypothesis as greater than (or less than) X1 which forces H0 to be a less than or equal to (or greater than or equal) statement. For whatever reason, that wasn't the above requirements for this udacity assignment.

[58]:
```
#from statsmodels.stats.proportion import proportions_ztest
#count = np.array([convert_old, convert_new])
#nobs = np.array([n_old, n_new])
#stat, pval = proportions_ztest(count, nobs)
#print('{0:0.16f}'.format(stat))
#print('{0:0.17f}'.format(pval))
```

[60]:
```
#This is amended per project reviewer's direction
#from statsmodels.stats.proportion import proportions_ztest:
#The alternative hypothesis can be either two-sided or one of the one- sided␣
 ↪tests,
#smaller means that the alternative hypothesis is prop < value and larger means␣
 ↪prop > value.
#In the two sample test, smaller means that the alternative hypothesis is p1 <␣
 ↪p2 and
#larger means p1 > p2 where p1 is the proportion of the first sample and p2 of␣
 ↪the second one.
from statsmodels.stats.proportion import proportions_ztest
count = np.array([convert_old, convert_new])
nobs = np.array([n_old, n_new])
stat, pval = proportions_ztest(count, nobs, alternative='smaller')
print('{0:0.16f}'.format(stat))
```

```
print('{0:0.17f}'.format(pval))
```

```
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.**?

The Z-score tells us that based on our initial data, the area outside the two tails is 1.3109. When that Z-Score to converted to a P-Value, the probablility is 0.18988. And, what does that mean? We can say that the P-value of .905 means that if the null hypothesis (H0 : Pold Conversion Rate = Pnew Conversion Rate) is true, we could expect 90 samples out of 100 to yield the results we obtained.
In other words, our observed results are not unusual. Because the P-Value (0.905) is greater than the level of significance we assigned (0.05), we fail to reject the null hypothesis and continue to accept that "old pager"'s converted rate is equal to the "new pager"'s converted rate .
Part III - A regression approach
`1.` In this final part, you will see that the result you acheived in the previous A/B test can also be acheived by performing regression.

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

Since we're seeking to predict rate of conversion (y-hat) based on the landing page (x), one might think of fitting a simple linear regression. However, two points make this an ideal candidate for fitting a logistic regression model. First, we're predicting categorical responses here as opposed to quantitative responses. Second, there are only two possible outcomes for any given landing page: conversion or not. Therefore, we'll fit a logistic regression model

b. The goal is to use **statsmodels** to fit the regression model you specified in part **a.** to see if there is a significant difference in conversion based on which page a customer receives. However, you first need to create a 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**.

[71]:
```
#never mentioned in lecture nor videos but necesary to import
from scipy import stats
stats.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df)

df2[['treatment', 'control']] = pd.get_dummies(df2['group'])
df2['ab_page'] = df2['treatment']
df2.head(10)
```

[71]:
```
   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
5   936923  2017-01-10 15:20:49.083499    control     old_page          0
6   679687  2017-01-19 03:26:46.940749  treatment     new_page          1
7   719014  2017-01-17 01:48:29.539573    control     old_page          0
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

   treatment  control  ab_page  intercept
0          1        0        1          1
1          1        0        1          1
2          0        1        0          1
3          0        1        0          1
4          1        0        1          1
5          1        0        1          1
6          0        1        0          1
7          1        0        1          1
8          0        1        0          1
9          0        1        0          1
```

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

[72]:
```
#never mentioned in lecture nor videos but necesary
from scipy import stats
stats.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df)
```

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

[73]:
```
df2['intercept'] = 1
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
```

[73]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              converted   No. Observations:               290584
Model:                          Logit   Df Residuals:                   290582
Method:                           MLE   Df Model:                            1
Date:                Mon, 01 Jun 2020   Pseudo R-squ.:               8.077e-06
```

```
Time:                        15:05:30   Log-Likelihood:            -1.0639e+05
converged:                       True   LL-Null:                   -1.0639e+05
                                        LLR p-value:                    0.1899
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -2.0038      0.008   -247.146      0.000      -2.020      -1.988
ab_page        0.0150      0.011      1.311      0.190      -0.007       0.037
==============================================================================
"""
```

In order to interpret this summary further, we need to exponentiate these results:

[74]:
```
np.exp(results.params)
```

[74]:
```
intercept    0.134827
ab_page      1.015102
dtype: float64
```

Converting is 1.015102 times as likely for those "new pagers" (treatment group) than for the "old pagers" (control group) holding all else constant.

In simpler terms, converting is almost just as likely for the "old pagers" (control group) as it is for the "new pagers (treatment group), holding all else constant.

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 the **Part II**?

INCREDIBLY IMPORTANT NOTE (revisited for a third time): it's a valuable reminder here that per the customer requirements, the hypothesis test in Part II, A/B testing, is being set up with the null hypothesis setting "old pagers" and "new pagers" converted rates as being EQUAL TO EACH OTHER and the alternative with them being NOT BEING EQUAL to each other. Why is this important to note while testing? Because it sets in motion that all levels of significance will be split evenly in a two-tailed test.

Based on my personal experience running hypothesis testing in my previous career, the majority of the customer requirements choose to show the alternative hypothesis as greater than (or less than) X1 which forces H0 to be a less than or equal to (or greater than or equal) statement. For whatever reason, that wasn't the above requirements for this udacity assignment.

In this Logistic Regression model, the P-value.190 again fails to reject the null in favor of the alternative. In part II above, again, P-value .1899 also fails to reject the null in favor of the alternative hypotheis (H0: Old page converted rates = New page converted rates). As I see from the hint, I'm inclined to believe (as I've noted above for the 3rd time), that Udacity indeed did want the analyst to use a less than or equal to statement in the null hypothesis in part II, A/B testing, rather than an equal statement as they indicated initially.

Noted by the analyst three times, this was not clearly stated in the customer requirements and so this analyst set the "old pagers" and "new pagers" equal to the converted rate and to

each other. The Logistic Regression model was supposed to showcase a difference in P-value from the P-value in Part II, A/B testing model which would have show a P-value difference between the two and would have been attributed to the change from a "less than/equal to" null hypothesis in Part II, A/B testing model to this null hypothesis which is essentially set as "equal to"

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?

Adding additional variables has the advantage of raising P-values thereby showing greater sigficance for explaining the variability for the model (and therefore, the Y-value) under display.

However, adding additional variables (at least in multiple linear regression) adds the disadvantage of the potential for collinearity between variables. Meaning, there is a level of dependence of one variable on another variable. There are ways to mitigate that effect but adding variables does require at least acknowledging that disadvantage to variable addition.

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

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

[75]:
```
countries_df = pd.read_csv('./countries.csv')
countries_df.head()
```

[75]:
```
   user_id country
0   834778      UK
1   928468      US
2   822059      UK
3   711597      UK
4   710616      UK
```

[76]:
```
df_new = countries_df.set_index('user_id').join(df2.set_index('user_id'),␣
 ↪how='inner')
```

The command to "join" sorts the two lists according to user_id.

[77]:
```
df_new.head()
```

[77]:
```
        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  treatment  control  ab_page  intercept
user_id
834778           0          1        0        1          1
928468           0          0        1        0          1
822059           1          0        1        0          1
711597           0          1        0        1          1
710616           0          0        1        0          1
```

[78]:
```python
### Create the necessary dummy variables
country_dummies = pd.get_dummies(df_new['country'])
df_countries = df_new.join(country_dummies)
df_countries.head()
```

[78]:
```
        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  treatment  control  ab_page  intercept  CA  UK  US
user_id
834778           0          1        0        1          1   0   1   0
928468           0          0        1        0          1   0   0   1
822059           1          0        1        0          1   0   1   0
711597           0          1        0        1          1   0   1   0
710616           0          0        1        0          1   0   1   0
```

[79]:
```python
#Will drop largest country (keeping it as baseline)
df_countries['country'].value_counts()
```

[79]:
```
US    203619
UK     72466
CA     14499
Name: country, dtype: int64
```

[84]:
```python
### Fit Your Linear Model And Obtain the Results
df_countries['intercept'] = 1
log_mod = sm.Logit(df_countries['converted'], df_countries[['intercept', 'UK',␣
 ↪'CA']])
results = log_mod.fit()
results.summary()
```

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

[84]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              converted   No. Observations:               290584
Model:                          Logit   Df Residuals:                   290581
Method:                           MLE   Df Model:                            2
Date:                Mon, 01 Jun 2020   Pseudo R-squ.:               1.521e-05
Time:                        15:16:51   Log-Likelihood:            -1.0639e+05
converged:                       True   LL-Null:                   -1.0639e+05
                                        LLR p-value:                    0.1984
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -1.9967      0.007   -292.314      0.000      -2.010      -1.983
UK             0.0099      0.013      0.746      0.456      -0.016       0.036
CA            -0.0408      0.027     -1.518      0.129      -0.093       0.012
==============================================================================
"""
```

[81]:
```
np.exp(results.params)
```

[81]:
```
intercept    0.135779
UK           1.009966
CA           0.960018
dtype: float64
```

Converting is 1.009966 times more likely from UK site users when compared to US site users, holding all other variables constant. However, converting is 0.039982 (1.0-0.0960018) less likely from CA site users when compared to US site users, holding all other variables 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.

[86]:
```
df_countries['UK_ab_page'] = df_countries['UK']*df_countries['ab_page']
df_countries['CA_ab_page'] = df_countries['CA']*df_countries['ab_page']
```

[87]:
```
### Fit Your Linear Model And Obtain the Results
df_countries['intercept'] = 1
log_mod = sm.Logit(df_countries['converted'], df_countries[['intercept', 'UK',␣
↪'CA']])
results = log_mod.fit()
results.summary()
```

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

[87]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              converted   No. Observations:               290584
Model:                          Logit   Df Residuals:                   290581
Method:                           MLE   Df Model:                            2
Date:                Mon, 01 Jun 2020   Pseudo R-squ.:               1.521e-05
Time:                        15:22:06   Log-Likelihood:            -1.0639e+05
converged:                       True   LL-Null:                   -1.0639e+05
                                        LLR p-value:                    0.1984
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -1.9967      0.007   -292.314      0.000      -2.010      -1.983
UK             0.0099      0.013      0.746      0.456      -0.016       0.036
CA            -0.0408      0.027     -1.518      0.129      -0.093       0.012
==============================================================================
"""
```

[88]:
```
np.exp(results.params)
```

[88]:
```
intercept    0.135779
UK           1.009966
CA           0.960018
dtype: float64
```

Converting is 1.009966 times more likely from UK site users when compared to US site users, holding all other variables constant. However, converting is 0.039982 (1.0-0.0960018) less likely from CA site users when compared to US site users, holding all other variables constant.

Optional:

In attempting to find a "goodness of fit" type indicator in Logistic regression (that one finds as "R2" in linear regression), the analyst noticed "Pseudo R-squ:" in the summary.

A quick document search https://stats.idre.ucla.edu/other/mult-pkg/faq/general/faq-what-are-pseudo-r-squareds/ showed that, though Pseudo R-squ was intended to fill that void within the logistic regression model, it has too many variabilities and therefore doesn't offer the opportunity to be an accurate indicator of fit. It just doesn't "fit" as well as R2'd fits for linear regression (pun intended).

Conclusion

Run through descriptive statistics as well as multiple tests in inferenctial statistics, the analysis has failed to reject the null hypothesis that the conversion rates between the "new pagers" and "old pagers" are anything but equal to each other. Therefore, is is the analyst's recommendation not to invest in launching a new website solely based on the notion that a new page will draw a higer customer conversion rate than the existing website.