3_Analyze_AB_Test_Results

May 20, 2019

1 Analyze A/B Test Results

Author: Ken Norton (ken@kennethnorton.com)

Table of Contents

Analyze A/B Test Results
Introduction
Part I - Probability
Answer (4.e.)
Part II - A/B Test
Answer (1.)
Answer (2.k.)
Answer (2.m.)
Part III - A regression approach
Answer (1.a.)
Answer (1.e.)
Answer (1.f.)
Answer (1.g.)
Answer (1.h.)
Extra: Looking at the Effect of Time

1.0.1 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.

1.0.2 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
        import statsmodels.api as sm

        %matplotlib inline
        %config InlineBackend.figure_format = 'retina'

        # We are setting the seed to assure you get
        # the same answers on quizzes as we set up
        random.seed(42)
```

```
In [2]: # PyPlot style sheets
        plt.style.use('fivethirtyeight')
        plt.style.use('seaborn-poster')
```

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 [3]: df = pd.read_csv('data/ab_data.csv')
        df.head()
```

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

```
In [4]: df.groupby('group')['landing_page'].value_counts()
```

```
Out[4]: group      landing_page
        control    old_page        145274
                   new_page          1928
        treatment  new_page        145311
                   old_page          1965
        Name: landing_page, dtype: int64
```

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

```
In [5]: df.describe()
```

```
Out[5]:             user_id      converted
        count  294478.000000  294478.000000
        mean   787974.124733       0.119659
        std     91210.823776       0.324563
        min    630000.000000       0.000000
        25%    709032.250000       0.000000
        50%    787933.500000       0.000000
        75%    866911.750000       0.000000
        max    945999.000000       1.000000
```

There are 29,4478 rows.

c. The number of unique users in the dataset.

```
In [6]: df.user_id.nunique()
```

```
Out[6]: 290584
```

d. The proportion of users converted.

```
In [7]: df['converted'].value_counts(normalize=True) * 100
```

```
Out[7]: 0    88.034081
        1    11.965919
        Name: converted, dtype: float64
```

11.97% of users converted.

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

```
In [8]: df.query('group == "treatment" and landing_page != "new_page"').count(
        ) + df.query('group != "treatment" and landing_page == "new_page"').count()
```

```
Out[8]: user_id         3893
        timestamp       3893
        group           3893
        landing_page    3893
        converted       3893
        dtype: int64
```

f. Do any of the rows have missing values?

```
In [9]: 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
```

No, 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**.

```
In [10]: df2 = df.query((
             '(group == "treatment" and landing_page == "new_page") \
              or (group == "control" and landing_page == "old_page")'
         ))
```

```
In [11]: # 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]
```

```
Out[11]: 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 [12]: df2.user_id.nunique()
```

```
Out[12]: 290584
```

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

```
In [13]: 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
```

We see there are 290,584 unique `user_id`s but 290,585 rows, so one `user_id` is duplicated. Let's find out which one.

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

```
In [14]: df2[df2.duplicated(subset=['user_id'],keep=False)]
```

```
Out[14]:       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
```

The repeated `user_id` is 773192.

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

```
In [15]: # Drop duplicates, keep first row
         df2 = df2.drop_duplicates(subset='user_id', keep='first')
```

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

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

```
In [16]: overall_rate = df2.query(
             'converted == 1').user_id.nunique() / df2.user_id.nunique()
         overall_rate
```

```
Out[16]: 0.11959708724499628
```

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

```
In [17]: control_df = df2.query('group == "control"')
```

```
In [18]: control_rate = control_df.query(
             'converted == 1').user_id.nunique() / control_df.user_id.nunique()
         control_rate
```

```
Out[18]: 0.1203863045004612
```

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

```
In [19]: treatment_df = df2.query('group == "treatment"')
```

```
In [20]: treatment_rate = treatment_df.query(
             'converted == 1').user_id.nunique() / treatment_df.user_id.nunique()
         treatment_rate
```

```
Out[20]: 0.11880806551510564
```

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

```
In [21]: df2.query(
             'landing_page == "new_page"').user_id.nunique() / df2.user_id.nunique()
```

```
Out[21]: 0.5000619442226688
```

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.

Answer (4.e.) There is not yet enough evidence to reject the null hypothesis that any differences in conversion between the two pages is due to chance.

1.0.3 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 (1.) Null hypothesis:

My null hypothesis is that the new page is no better, or possibly even worse, than the old version. Expressed as:

$H_0 : P_{new} \leq P_{old}$

Alternative hypothesis:

My alternative hypothesis is that the new page *is* better than the old version. Expressed as:

$H_0 : P_{new} > P_{old}$

`2.` Assume under the null hypothesis, p_{new} and p_{old} both have "true" success rates equal to the **converted** success rate regardless of page - that is p_{new} and p_{old} are equal. Furthermore, assume they are equal to the **converted** rate in **ab_data.csv** regardless of the page.

Use a sample size for each page equal to the ones in **ab_data.csv**.

Perform the sampling distribution for the difference in **converted** between the two pages over 10,000 iterations of calculating an estimate from the null.

Use the cells below to provide the necessary parts of this simulation. If this doesn't make complete sense right now, don't worry - you are going to work through the problems below to complete this problem. You can use **Quiz 5** in the classroom to make sure you are on the right track.

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

```
In [22]: # It's equivalent to the overall conversion rate
         p_new = overall_rate
         p_new
```

```
Out[22]: 0.11959708724499628
```

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

```
In [23]: # It's equivalent to the overall conversion rate
         p_old = overall_rate
         p_old
```

```
Out[23]: 0.11959708724499628
```

c. What is n_{new}?

```
In [24]: n_new = treatment_df.user_id.nunique()
         n_new
```

```
Out[24]: 145310
```

d. What is n_{old}?

```
In [25]: n_old = control_df.user_id.nunique()
         n_old
```

```
Out[25]: 145274
```

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

```
In [26]: new_page_converted = np.random.choice(
             [0, 1], n_new, p=((1 - overall_rate), overall_rate))
         new_page_converted
```

```
Out[26]: array([0, 0, 0, ..., 0, 0, 1])
```

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

```
In [27]: old_page_converted = np.random.choice(
             [0, 1], n_old, p=((1 - overall_rate), overall_rate))
         old_page_converted
```

```
Out[27]: array([0, 0, 1, ..., 0, 0, 0])
```

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

```
In [28]: new_page_converted.mean() - old_page_converted.mean()
```

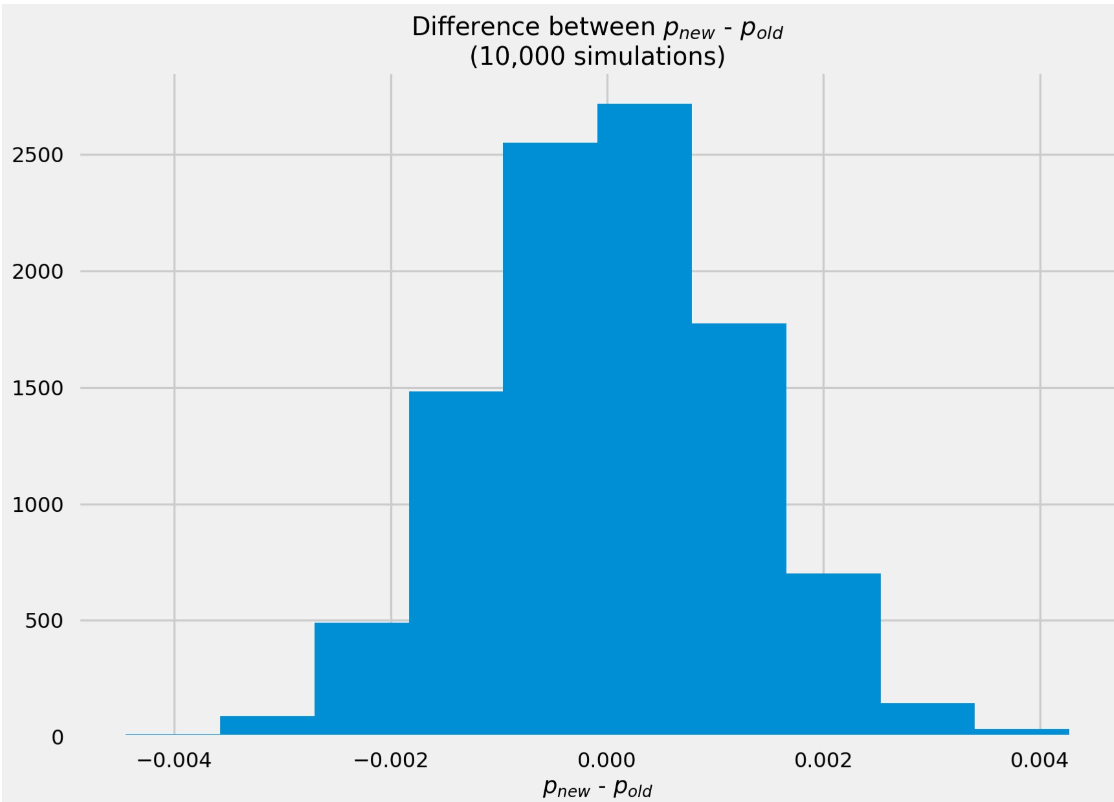
```
Out[28]: -0.00033923172618709196
```

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**.

```
In [29]: p_diffs = []

         for _ in range(10000):
             new_page_converted = np.random.choice([0, 1],
                                                   n_new,
                                                   p=((1 - overall_rate), overall_rate))
             old_page_converted = np.random.choice([0, 1],
                                                   n_old,
                                                   p=((1 - overall_rate), overall_rate))
             p_diffs.append(new_page_converted.mean() - old_page_converted.mean())

         p_diffs = np.asarray(p_diffs)
```

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 [30]: plt.hist(p_diffs)
         plt.title("Difference between $p_{new}$ - $p_{old}$\n(10,000 simulations)")
         plt.xlabel("$p_{new}$ - $p_{old}$")
         plt.show()
```



The plot works as we expected with a standard distribution.

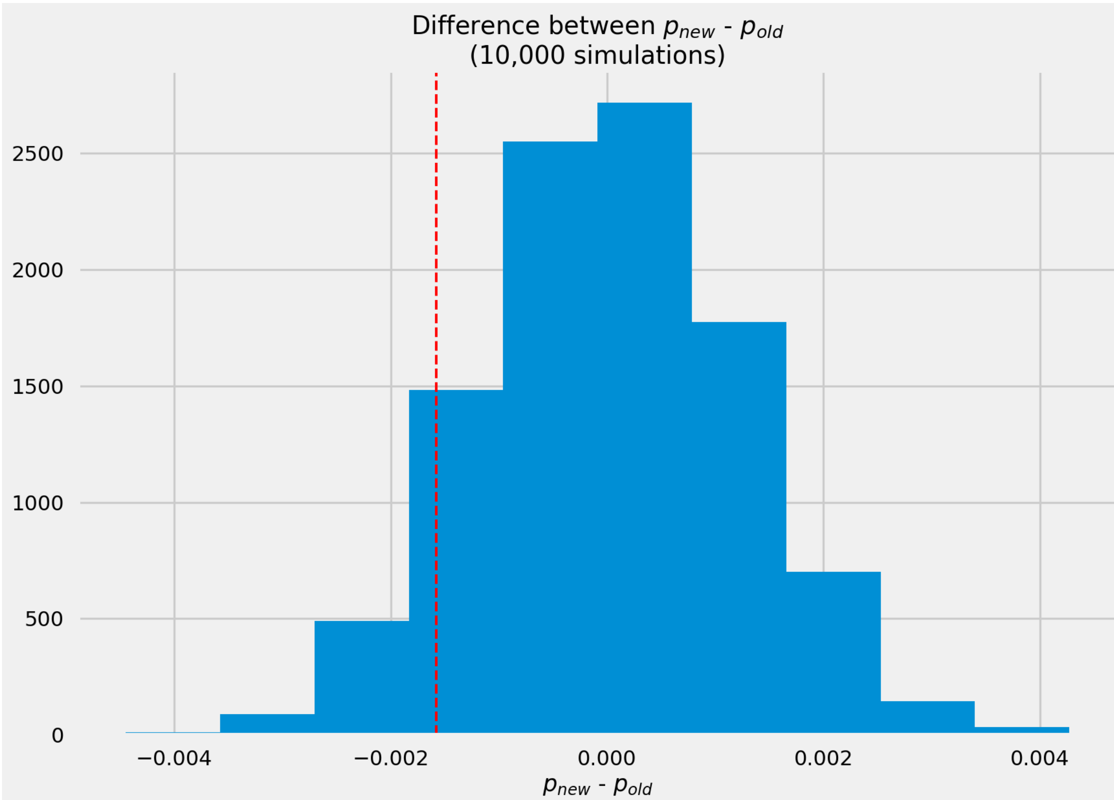
j. What proportion of the **p_diffs** are greater than the actual difference observed in **ab_data.csv**?

```
In [31]: obs_diff = (treatment_rate - control_rate)

         print("obs_diff: ", obs_diff)
         print("p_diffs > obs_diff: ", (p_diffs > obs_diff).mean())
```

```
obs_diff:  -0.0015782389853555567
p_diffs > obs_diff:  0.91
```

```
In [32]: plt.hist(p_diffs)
         plt.title("Difference between $p_{new}$ - $p_{old}$\n(10,000 simulations)")
```

```
plt.xlabel("$p_{new}$ - $p_{old}$")
plt.axvline(obs_diff, color='r', linestyle='dashed', linewidth=2)
plt.show()
```



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?

Answer (2.k.) I just calculated the p-value, which is 0.9. That means that in a chance model, the results of our experiment are reproduced 90% of the time. That clearly indicates that we have failed to reject our null hypothesis and find $H_0 : P_{new} \leq P_{old}$

In order to accept the alternative hypothesis, we'd want to an α (alpha) of 0.05 or below.

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

```
convert_old = df2.query(
    'converted == 1 and landing_page == "old_page"').user_id.nunique()
convert_new = df2.query(
    'converted == 1 and landing_page == "new_page"').user_id.nunique()
n_old = df2.query('landing_page == "old_page"').user_id.nunique()
n_new = df2.query('landing_page == "new_page"').user_id.nunique()
```

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

Note: the link above is broken but I was able to find a cached version of the page at the Internet Archive

```
In [34]: # The function receives (count, nobs, alternative) where
         # count and nobs are arrays representing the two trials
         zstat, pval = sm.stats.proportions_ztest([convert_old, convert_new],
                                                  [n_old, n_new],
                                                  alternative='larger')

         print("z-stat: ", zstat, "\np-value: ", pval)

z-stat:  1.3109241984234394
p-value:  0.09494168724097551
```

We can use scipy to see if the z-score is significant

```
In [35]: from scipy.stats import norm

         # Tells us how significant our z-score is
         norm.cdf(zstat)

Out[35]: 0.9050583127590245

In [36]: # Tells us what our critical value at 95% confidence is
         norm.ppf(1-(0.05/2))

Out[36]: 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.**?

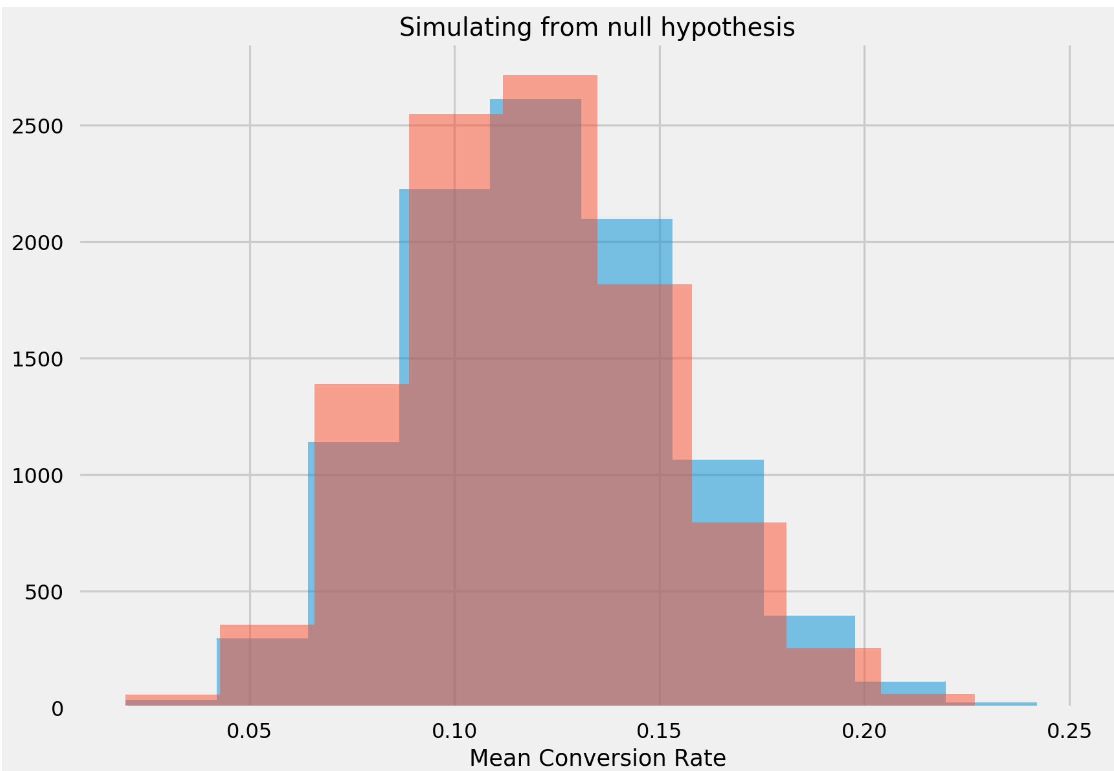
Answer (2.m.) Since the z-score of 0.90 does not exceed the critical value at 95% confidence (1.96) we fail to reject the null hypothesis. Our conclusion agrees with the findings in j. and k. above.

Alternate approach: bootstrap simulating from the null hypothesis

Another approach is to simulate from the null hypothesis, as shown in Lesson 12. Here we bootstrap the sample data from our entire result set and take the mean of the new_page, mean of the old_page, and the mean difference between the two. We'll run this over 10,000 iterations.

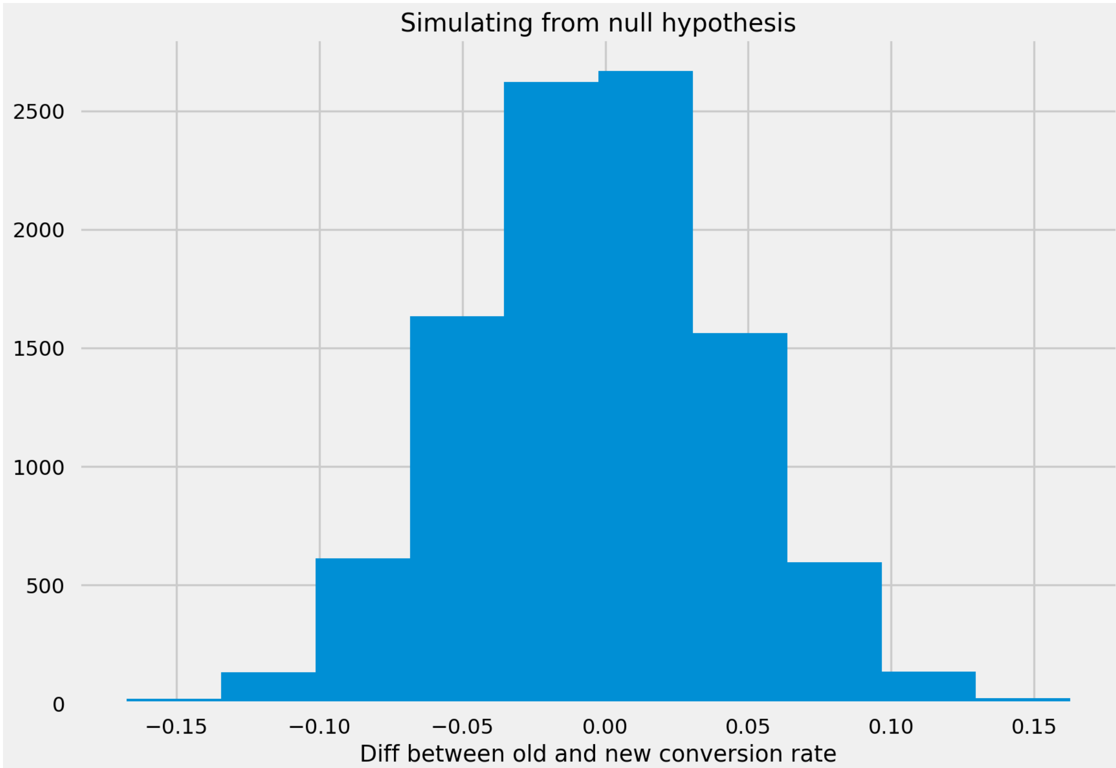
```
In [37]: old_means, new_means, diffs = [], [], []

         for _ in range(10000):
             bootsamp = df2.sample(200, replace=True)
             new_mean = bootsamp[bootsamp['landing_page'] ==
                                 "new_page"]['converted'].mean()
             old_mean = bootsamp[bootsamp['landing_page'] ==
                                 "old_page"]['converted'].mean()

             new_means.append(new_mean)
             old_means.append(old_mean)
             diffs.append(new_mean - old_mean)
```

```
In [38]: plt.hist(old_means, alpha = 0.5)
         plt.hist(new_means, alpha = 0.5)
         plt.title("Simulating from null hypothesis")
         plt.xlabel("Mean Conversion Rate")
         plt.show()
```



The two treatments appear to form a standard distribution around the same point.

```
In [39]: plt.hist(diffs)
         plt.title("Simulating from null hypothesis")
         plt.xlabel("Diff between old and new conversion rate")
         plt.show()
```



Here we can see that the difference between the two means follows a standard distribution around zero. This is exactly what we'd expect from the Central Limit Theorem and we have failed to reject our null hypothesis.

1.0.4 Part III - A regression approach

`1.` In this final part, you will see that the result you achieved in the previous A/B test 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 (1.a.) A logistic regression because our outcome (`converted`) is binary.

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**.

```
In [40]: df2['intercept'] = 1
         df2.loc[df2['group'] == 'treatment', 'ab_page'] = 1
         df2.loc[df2['group'] == 'control', 'ab_page'] = 0
```

```
In [41]: df2.head()
```

```
Out[41]:    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      0.0
         1          1      0.0
         2          1      1.0
         3          1      1.0
         4          1      0.0
```

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

```
In [42]: # Logistic regression
         logit = 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 [43]: # Logit regression
         results = logit.fit()
         results.summary()
```

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

```
Out[43]: <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, 20 May 2019   Pseudo R-squ.:               8.077e-06
         Time:                        13:58:31   Log-Likelihood:            -1.0639e+05
         converged:                       True   LL-Null:                   -1.0639e+05
                                                 LLR p-value:                    0.1899
         ==============================================================================
                          coef    std err          z      P>|z|      [0.025      0.975]
         ------------------------------------------------------------------------------
```

```
intercept     -1.9888      0.008   -246.669      0.000      -2.005      -1.973
ab_page       -0.0150      0.011     -1.311      0.190      -0.037       0.007
==============================================================================
"""
```

e. What is the p-value associated with **ab_page**? Why does it differ from the value you found in **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**?

Answer (1.e.) The p-value is p=0.19. This is less than p=0.9 from Part II above. The reason for this is we are comparing two different null hypotheses. This was a two-tailed test, whereas our test in Part I was a one-tailed test (only testing whether our experiment group conversions *exceeded* the null hypothesis conversions). Here's how that's expressed in H-notation:

In Part II, our null and alternative hypotheses were (one-tailed test):
$H_0 : P_{new} \leq P_{old}$
$H_1 : P_{new} > P_{old}$
In this section, our null and alternative hypotheses are (two-tailed):
$H_0 : P_{new} = P_{old}$
$H_1 : P_{new} \neq P_{old}$
Despite the differences, we still fail to reject the null hypothesis because our p-value of 0.19 is above our α (alpha) of 0.05.

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 (1.f.) Adding other factors to a regression model (multiple regression) can help us understand the relative influence of several factors on conversion. There are disadvantages to multiple regression: while we may begin to understand relationships between factors, we have to be careful about drawing *causality* conclusions. Also, linear regression is sensitive to outliers. Adding more factors increases the likelihood that we introduce more outliers.

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.

In [44]:
```python
countries_df = pd.read_csv('data/countries.csv')
```

In [45]:
```python
countries_df['country'].unique()
```

Out[45]:
```
array(['UK', 'US', 'CA'], dtype=object)
```

In [46]:
```python
df_new = countries_df.set_index(
    'user_id').join(df2.set_index('user_id'), how='inner')
df_new.head()
```

Out[46]:

```
        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      0.0
928468           0          1      1.0
822059           1          1      1.0
711597           0          1      0.0
710616           0          1      1.0
```

In [47]:

```python
# Create dummy columns for country, join them to df_new,
# drop the country column
country_dummies = pd.get_dummies(df_new['country'])
df_new = df_new.join(country_dummies)
df_new.drop(columns = ['country'], inplace=True)
df_new.head()
```

Out[47]:

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

         intercept  ab_page  CA  UK  US
user_id
834778           1      0.0   0   1   0
928468           1      1.0   0   0   1
822059           1      1.0   0   1   0
711597           1      0.0   0   1   0
710616           1      1.0   0   1   0
```

In [48]:

```python
mlr = sm.Logit(df_new['converted'],
               df_new[['intercept', 'CA', 'UK']])
results_mlr = mlr.fit()
results_mlr.summary()
```

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

```
Out[48]: <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, 20 May 2019   Pseudo R-squ.:               1.521e-05
Time:                        13:58:31   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
CA            -0.0408      0.027     -1.518      0.129      -0.093       0.012
UK             0.0099      0.013      0.746      0.456      -0.016       0.036
==============================================================================
"""
```

Answer (1.g.) Since the p-values for CA and UK vs. US are p=0.13 and p=0.46 respectively, there is no evidence to reject the null hypothesis that country has no impact on conversion.

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 [49]: mlr = sm.Logit(df_new['converted'],
                        df_new[['intercept', 'ab_page', 'CA', 'UK']])
         results_mlr = mlr.fit()
         results_mlr.summary()
```

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

```
Out[49]: <class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              converted   No. Observations:               290584
Model:                          Logit   Df Residuals:                   290580
Method:                           MLE   Df Model:                            3
Date:                Mon, 20 May 2019   Pseudo R-squ.:               2.323e-05
Time:                        13:58:32   Log-Likelihood:            -1.0639e+05
```

```
converged:                           True   LL-Null:                   -1.0639e+05
                                            LLR p-value:                    0.1760
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -1.9893      0.009   -223.763      0.000      -2.007      -1.972
ab_page       -0.0149      0.011     -1.307      0.191      -0.037       0.007
CA            -0.0408      0.027     -1.516      0.130      -0.093       0.012
UK             0.0099      0.013      0.743      0.457      -0.016       0.036
==============================================================================
"""
```

Answer (1.h.) Since the p-values for ab_page, CA and UK are p=0.19, p=0.13 and p=0.74 respectively, there is no evidence to reject the null hypothesis that country has no impact on conversion. None of these p-values exceed our α of 0.05.

1.0.5 Extra: Looking at the Effect of Time

For extra credit I decided to look at the impact of time on conversion. Unfortunately it's difficult to make assumptions about the time of day for any given user since they are distributed across the globe, or at least in UK vs. CA/US. So rather than make assumptions about the time of day I decided to look at whether or not it was a weekday or the weekend. I still have some time zone considerations here (e.g. Monday morning UTC is still the weekend in US/CA).

```
In [50]: df_new.dtypes
```

```
Out[50]: timestamp        object
         group            object
         landing_page     object
         converted         int64
         intercept         int64
         ab_page         float64
         CA                uint8
         UK                uint8
         US                uint8
         dtype: object
```

First I'll need to convert the `timezone` column to a datetime.

```
In [51]: df_new['timestamp'] = pd.to_datetime(df_new['timestamp'] )
```

```
In [52]: df_new.dtypes
```

```
Out[52]: timestamp       datetime64[ns]
         group                   object
         landing_page            object
         converted                int64
         intercept                int64
         ab_page                float64
```

```
CA                        uint8
UK                        uint8
US                        uint8
dtype: object
```

Now I'll use Pandas datetime built-in to add a weekday column. This is a value from 0-6 where Monday is 0 and Sunday is 6.

```
In [53]: df_new['weekday'] = df_new['timestamp'].dt.weekday
```

```
In [54]: df_new.head()
```

```
Out[54]:                          timestamp      group landing_page  converted  \
         user_id
         834778  2017-01-14 23:08:43.304998    control     old_page          0
         928468  2017-01-23 14:44:16.387854  treatment     new_page          0
         822059  2017-01-16 14:04:14.719771  treatment     new_page          1
         711597  2017-01-22 03:14:24.763511    control     old_page          0
         710616  2017-01-16 13:14:44.000513  treatment     new_page          0

                  intercept  ab_page  CA  UK  US  weekday
         user_id
         834778           1      0.0   0   1   0        5
         928468           1      1.0   0   0   1        0
         822059           1      1.0   0   1   0        0
         711597           1      0.0   0   1   0        6
         710616           1      1.0   0   1   0        0
```

```
In [55]: # After creating my own function and testing some different approaches
         # I found this quick method to convert dayofweek to a 0 or 1 from
         # StackOverflow:
         #
         # https://stackoverflow.com/questions/32278728/convert- \
         # dataframe-date-row-to-a-weekend-not-weekend-value
         #
         # It simply tests to see if the value is less than or greater
         # than 5 (Mon-Fri)
         df_new['weekend'] = (df_new['timestamp'].dt.dayofweek // 5 == 1).astype(int)
```

```
In [56]: # Test to make sure it worked as expected
         df_new.head(10)
```

```
Out[56]:                          timestamp      group landing_page  converted  \
         user_id
         834778  2017-01-14 23:08:43.304998    control     old_page          0
         928468  2017-01-23 14:44:16.387854  treatment     new_page          0
         822059  2017-01-16 14:04:14.719771  treatment     new_page          1
         711597  2017-01-22 03:14:24.763511    control     old_page          0
         710616  2017-01-16 13:14:44.000513  treatment     new_page          0
```

```
909908  2017-01-06 20:44:26.334764  treatment     new_page          0
811617  2017-01-02 18:42:11.851370  treatment     new_page          1
938122  2017-01-10 09:32:08.222716  treatment     new_page          1
887018  2017-01-06 11:09:40.487196  treatment     new_page          0
820683  2017-01-14 11:52:06.521342  treatment     new_page          0

         intercept  ab_page  CA  UK  US  weekday  weekend
user_id
834778           1      0.0   0   1   0        5        1
928468           1      1.0   0   0   1        0        0
822059           1      1.0   0   1   0        0        0
711597           1      0.0   0   1   0        6        1
710616           1      1.0   0   1   0        0        0
909908           1      1.0   0   1   0        4        0
811617           1      1.0   0   0   1        0        0
938122           1      1.0   0   0   1        1        0
887018           1      1.0   0   0   1        4        0
820683           1      1.0   0   0   1        5        1
```

In [57]:
```
# Now I can drop the weekday column since I won't be using it
df_new.drop(columns = ['weekday'], inplace=True)
```

In [58]:
```
df_new.head()
```

Out[58]:
```
                          timestamp      group landing_page  converted  \
user_id
834778  2017-01-14 23:08:43.304998    control     old_page          0
928468  2017-01-23 14:44:16.387854  treatment     new_page          0
822059  2017-01-16 14:04:14.719771  treatment     new_page          1
711597  2017-01-22 03:14:24.763511    control     old_page          0
710616  2017-01-16 13:14:44.000513  treatment     new_page          0

         intercept  ab_page  CA  UK  US  weekend
user_id
834778           1      0.0   0   1   0        1
928468           1      1.0   0   0   1        0
822059           1      1.0   0   1   0        0
711597           1      0.0   0   1   0        1
710616           1      1.0   0   1   0        0
```

In [59]:
```
mlr = sm.Logit(df_new['converted'],
               df_new[['intercept', 'ab_page', 'CA', 'UK', 'weekend']])
results_mlr = mlr.fit()
results_mlr.summary()
```

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

Out[59]:
```
<class 'statsmodels.iolib.summary.Summary'>
"""
                           Logit Regression Results
==============================================================================
Dep. Variable:              converted   No. Observations:               290584
Model:                          Logit   Df Residuals:                   290579
Method:                           MLE   Df Model:                            4
Date:                Mon, 20 May 2019   Pseudo R-squ.:               2.324e-05
Time:                        13:58:32   Log-Likelihood:            -1.0639e+05
converged:                       True   LL-Null:                   -1.0639e+05
                                        LLR p-value:                    0.2929
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
intercept     -1.9894      0.010   -208.115      0.000      -2.008      -1.971
ab_page       -0.0149      0.011     -1.307      0.191      -0.037       0.007
CA            -0.0408      0.027     -1.516      0.130      -0.093       0.012
UK             0.0099      0.013      0.743      0.457      -0.016       0.036
weekend        0.0006      0.013      0.045      0.964      -0.025       0.026
==============================================================================
"""
```

When I add weekend to the multiple regression, I see that it has a p-value of 0.964 which is above our α of 0.05. The effect of weekends on conversion is not significant. I am sure there are other dummy variables we could test that look at time of day (e.g. morning, afternoon, evening) but that would require dealing with timezones to make sure the conversion is appropriate for the user's country.

In [ ]: