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!

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 (https://review.udacity.com/#!/projects/37e27304-ad47-4eb0-a1ab-8c12f60e43d0/rubric).

Part I - Probability

To get started, let's import our libraries.

In [291]:
```
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
import statsmodels.api as sm
%matplotlib inline
random.seed(42) # set seed to assure same results every time you run the
 code and to line up with quizzes
```

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 [292]:
```
df = pd.read_csv('ab_data.csv')
df.head()
```

Out[292]:

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

In [293]:
```
df.shape
```

Out[293]: (294478, 5)

The dataset has 294,478 rows and 5 columns

c. The number of unique users in the dataset.

In [294]:
```
df['user_id'].nunique()
```

Out[294]: 290584

The dataset has 290,584 unique users

d. The proportion of users converted.

In [295]:
```
df.converted.mean()
```

Out[295]: 0.11965919355605512

Proportion of users converted (proportion of 1s to 0s in the 'converted' column): 11.97%

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

There are 2 different ways that new_page and treatment don't line up:

- Treatment group, old_page
- Control group, new_page

In [296]:
```
treatment_old = df.query('group == "treatment" and landing_page == "old_
page"').nunique()
control_new = df.query('group == "control" and landing_page == "new_pag
e"').nunique()
treatment_old, control_new, treatment_old + control_new
```

Out[296]:
```
(user_id         1965
 timestamp       1965
 group              1
 landing_page       1
 converted          2
 dtype: int64, user_id         1928
 timestamp       1928
 group              1
 landing_page       1
 converted          2
 dtype: int64, user_id         3893
 timestamp       3893
 group              2
 landing_page       2
 converted          4
 dtype: int64)
```

Treatment, old = 1965
Control, new = 1928
Number of times new_page and treatment don't line up: 1965+1928=3893

In [297]:
```
# much better way
df[((df['group'] == 'treatment') == (df['landing_page'] == 'new_page'))
== False].shape[0]
```

Out[297]:
```
3893
```

f. Do any of the rows have missing values?

In [298]:
```
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
```

There are no missing values (294,478 values for each column)

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**.

Note: If there were 294,478 rows in the df, df2 should have 3893 less rows = 290,585

In [299]:
```
# keep only the rows where treatment group and new landing page are aligned
# this will automatically also drop rows where new landing page & control group are aligned
df2 = df[((df['group'] == 'treatment') == (df['landing_page'] == 'new_page'))]

# check to make sure there are 290,585 rows
df2.shape[0]
```

Out[299]: 290585

In [300]:
```
# 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[300]: 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 [301]:
```
df2['user_id'].nunique()
```

Out[301]: 290584

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

In [302]:
```
sum(df2['user_id'].duplicated())
```

Out[302]: 1

In [303]:
```
# alternate way, but takes a long time:
# pd.concat(g for _, g in df2.groupby('user_id') if len(g) > 1)

ids = df2["user_id"]
df2[ids.isin(ids[ids.duplicated()])].sort_values("user_id")
```

Out[303]:

	user_id	timestamp	group	landing_page	converted
1899	773192	2017-01-09 05:37:58.781806	treatment	new_page	0
2893	773192	2017-01-14 02:55:59.590927	treatment	new_page	0

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

Rows 1899 and 2893: treatment group, new page, not converted

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

In [304]:
```
df2.drop([2893], inplace=True)
df2.shape[0]
```

```
/Users/karenbevis/anaconda3/lib/python3.6/site-packages/ipykernel/__mai
n__.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-d
ocs/stable/indexing.html#indexing-view-versus-copy
  if __name__ == '__main__':
```

Out[304]: 290584

In [305]:
```
# ensure no duplicates, result should now be 0
sum(df2['user_id'].duplicated())
```

Out[305]: 0

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 [306]:
```
df2.head()
```

Out[306]:

	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 [307]:
```
# calculate percentage of 1s to 0s in converted column
df2['converted'].mean()
```

Out[307]: 0.11959708724499628

In [308]:
```
# another check to ensure accuracy
df2.query('converted == 1').count()[0] / df2.shape[0]
```

Out[308]: 0.11959708724499628

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

In [309]:
```
df2.query('group == "control"')['converted'].mean()
```

Out[309]: 0.1203863045004612

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

In [310]:
```
df2.query('group == "treatment"')['converted'].mean()
```

Out[310]: 0.11880806551510564

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

In [311]:
```
df2.query("landing_page == 'new_page'").count()[0]/df2.shape[0]
```

Out[311]: 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.

Overall conversions: 11.96%
Control (old page) conversions: 12.04% (.08% increase)
Treatment (new page) conversions: 11.88% (.08% decrease)

No, there is not sufficient evidence to say that the new page leads to more conversions. The treatment showed a decrease in conversions by .08% while the control group increased by .08%. This difference is small, and we might benefit from more testing, however at this time it appears that the old page is more effective and the new page is not worth allocating resources. Both groups received the pages equally (~50% each).

```
In [312]: df['timestamp'].min(), df['timestamp'].max()
```

```
Out[312]: ('2017-01-02 13:42:05.378582', '2017-01-24 13:41:54.460509')
```

The test has been run for 22 days, which seems like an adequate amount of time (especially considering we tested more than 290,000 unique users).

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.

H_0: The new page is worse, or only as good, as the old.
Null: what we assume to be true by default.
p_{new} - **p_{old}** <= 0
H_1: The new page is better than the old.
Alternative: what we want to prove to be true.
p_{new} - **p_{old}** > 0

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.

Analyze A/B Test Steps:

- Compute OBSERVED DIFFERENCE between the metric (actual conversion rates for old and new pages, Part 1)
- Create a SAMPLING DISTRIBUTION (simulated samples) for the observed statistic, calculate simulated difference (single observation)
- Use the sampling distribution to SIMULATE DISTRIBUTION UNDER THE NULL (use larger sample size, find probability that the observed statistic came from this distribution)
- Compute the P-VALUE (find the proportion of values in the null distribution that were greater than our observed difference)

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

In [247]:
```
# p_new = p_old = overall conversion rate, 11.96%
p_new = df2.converted.mean()
p_new
```

Out[247]: 0.11959708724499628

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

In [248]:
```
# p_new = p_old = overall conversion rate, 11.96%
p_old = df2.converted.mean()
p_old
```

Out[248]: 0.11959708724499628

From Slack, Randy J: "Count the number in each group (n_old and n_new)."
Note to Udacity: I had no idea what n was, this should be explaned.

c. What is n_{new}?

In [249]:
```
n_new = df2.query("landing_page == 'new_page'")['converted'].count()
n_new
```

Out[249]: 145310

d. What is n_{old}?

In [250]:
```
n_old = df2.query('landing_page == "old_page"')['converted'].count()
n_old
```

Out[250]: 145274

Create simulated samples with random.binomial using the counts and proportions we determined from our actual data as parameters. Subtract the means of the results to get the observed (simulated) difference.

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 [326]:
```
# np.random.binomial(n, p, size)
# n=1 trial size, p=probabiity of trial, size=number of trials to run

new_page_converted = np.random.binomial(1, p_new, n_new)
new_page_converted.mean()
```

Out[326]: 0.11952377675314844

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 [338]:
```
old_page_converted = np.random.binomial(1, p_old, n_old)
old_page_converted.mean()
```

Out[338]: 0.1206065779148368

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

In [339]:
```
obs_simulation_diff = new_page_converted.mean() - old_page_converted.mea
n()
print('Our observed simulated difference is: {}'.format(obs_simulation_d
iff.mean()))
```

```
Our observed simulated difference is: -0.0010828011616883515
```

Is the difference in our simulated sample static significant or just due to chance? It's just one test. Run it 10,000 times.

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 [350]:
```
new_converted_simulation = np.random.binomial(n_new, p_new,  10000)/n_ne
w
old_converted_simulation = np.random.binomial(n_old, p_old,  10000)/n_ol
d
p_diffs = new_converted_simulation - old_converted_simulation
p_diffs = np.array(p_diffs)
old_converted_simulation.mean(), new_converted_simulation.mean()
```

Out[350]:
```
(0.11959352740338945, 0.11960109077145414)
```

In [255]:
```
# Alternative: Use loop to create a sampling distribution for our observ
ed statistic with binomials or bootstrapping

#p_diffs = []
#for _ in range(10000):
#    new_page_converted = np.random.binomial(1, p_new, n_new)
#    old_page_converted = np.random.binomial(1, p_old, n_old)
#    p_diffs.append(new_page_converted.mean() - old_page_converted.mean
())

# convert to numpy array to plot
#p_diffs = np.array(p_diffs)

# p_diffs = []
# for _ in range(10000):
#    boot_sample = df2.sample(df2.shape[0], replace = True)
#    new_page_converted = boot_sample.query('landing_page == "new_pag
e"')['converted'].mean()
#    old_page_converted = boot_sample.query('landing_page == "old_pag
e"')['converted'].mean()
#    p_diffs.append(new_page_converted - old_page_converted)
```

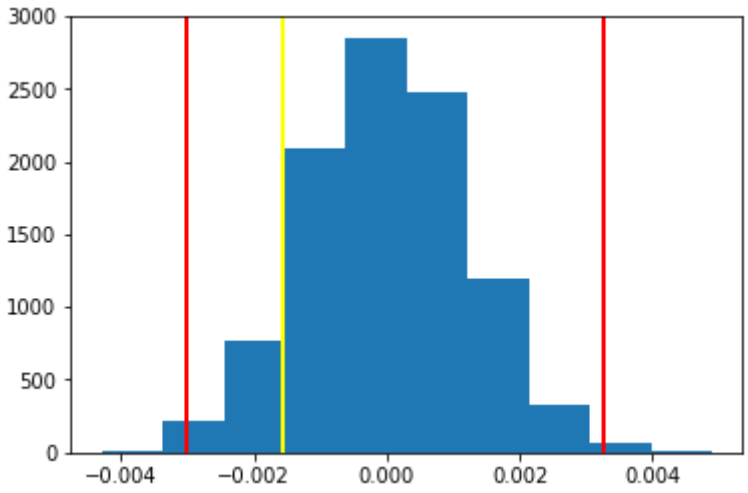
i. Plot a histogram of the **p_diffs**. Does this plot look like what you expected? Use the matching problem in the classroom to assure you fully understand what was computed here.

In [351]:
```
control_convert_actual = df2.query('group == "control"')['converted'].mean()
treatment_convert_actual =  df2.query('group == "treatment"')['converted'].mean()
obs_diff = treatment_convert_actual - control_convert_actual
```

In [352]:
```
p_diffs = np.array(p_diffs)

# view 99% confidence interval
low, upper = np.percentile(p_diffs, .5), np.percentile(p_diffs, 99.5)
low, upper

plt.hist(p_diffs);
plt.axvline(x=low, color='red', linewidth=2);
plt.axvline(x=upper, color='red', linewidth=2);
plt.axvline(obs_diff, color='yellow', linewidth=2);
```



Yes, according to the **Central Limit Theorem**, with a large enough sample size the sample mean follows a normal distribution (bell shaped) as expected. Also, looking at the 99% **Confidence Interval**, most data points fall within this region suggesting that our null hypothesized value did generate our statistic.

Next we'll calculate the probability (p-value) that our ovserved statistic came from this distrubution to see if it supports rejecting our null hypothesis.

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

In [353]:
```
#null value not needed here to calculate p-value
#null_mean = 0
#null_vals = np.random.normal(null_mean, np.std(p_diffs), len(p_diffs))

# find proportion of values greater than observed diff since since alter
native is > null.
print("Actual observed difference (sample statistic): {}".format(obs_dif
f))
print("Proportion of p_diffs greater than observed (p-value): {}".format
((p_diffs > obs_diff).mean()))
```

```
Actual observed difference (sample statistic): -0.0015782389853555567
Proportion of p_diffs greater than observed (p-value): 0.9057
```

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?

Our statistic appears to come from this distribution.

Since our alternative hypothesis is that p_new > p_old, the probability that our observed statistic comes from this distribution is the shading from the observed statistic (yellow line) to the RIGHT. This is more than half of our distribution, suggesting that our observed statistic does fall within this range and supporting our null hypothesis that the old page converts better (or equally the same) than the new.

Calculating p-value helps us to determine statistical significance of the observed difference. The p-value is the probability of observing our statistic, or more extreme values in favor of the alternative (greater in this case), if the null is true. A small p-value (small probability) means it is unlikely the we will observe our statistic fom the null, and more likely it came from the alternative.

P-value

Type I error: Deciding the new page (alt) is better, but really the old page (null) is better (worst error).
Type II error: Deciding the old page (null) is better, but really the new page (alt) is better.
Type I error threshold: 0.05

If p-value <= 0.05 (small): strong evidence against the null (reject the null)
If p-value > 0.05 (large): weak evidence against the null (fail to reject the null)

Since the p-value is large (~0.9), we do NOT have statistically significent evidence that suggests the new page converts better. Therefore, we **fail to reject the null** (stick with the old page).

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.

proportions_ztest is a function to check if 2 proportions are statistically better than (one-tail), worse than (one tail), or different from (two-tail). The default parameter for proportions_ztest is testing for different proportions (two-tailed, alternative='two-sided') and assumes:

Null: p_new - p_old = 0 (no difference between two proportions)
Alt: p_new - p_old !=0 (a statistical difference between two proportions)

We need to modify the function parameter 'alternative' from 'two-sided' to 'smaller' to achieve our alt hypotheses of new > old.

In [354]:
```
import statsmodels.api as sm

convert_old = df2.query("landing_page == 'old_page' and converted == 1")
.shape[0]
convert_new = df2.query("landing_page == 'new_page' and converted == 1")
.shape[0]
n_old = df2.query("landing_page == 'old_page'").shape[0]
n_new = df2.query("landing_page == 'new_page'").shape[0]
```

m. Now use `stats.proportions_ztest` to compute your test statistic and p-value. Here (http://knowledgetack.com/python/statsmodels/proportions_ztest/) is a helpful link on using the built in.

In [355]:
```
p_diffs.mean(), p_diffs.std(), p_diffs.var()
```

Out[355]:
```
(7.563368064675463e-06, 0.0012208420959154934, 1.4904554231593349e-06)
```

In [356]:
```
z_score, p_value = sm.stats.proportions_ztest([convert_old, convert_new
], [n_old, n_new], value=None, alternative='smaller', prop_var=False)
z_score, p_value
```

Out[356]:
```
(1.3109241984234394, 0.9050583127590245)
```

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.**?

P-value is a probability that we have falsely rejected the null hypothesis. Z-score measures standard deviation (the number of standard deviations from the mean a data point is), another test of statistical significance that helps us decide whether or not to reject the null. When the z-score is very large (or small) and associated with a very small p-value, this indicates it is unlikely that the pattern observed is a theoretically random pattern represented by the null (strong evidence against the null). Since the standard deviation here is small but our p-value is quite large, it does not appear to be of interest.

The large p-value (matching our earlier calculation), suggests that converstions from the old page are statistically better than the new.

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?

Regression Types/ Models:

- Simple Linear (SLR): compare 2 quantitative variables - use one explanatory variable (x) to predict a response variable (y)
- Multiple Linear (MLR): compare multiple explanatory variables (X) to predict a response variable (y) (predicted response not constrained, any value between negative and positive infinity)
- Logistic: predict categorical data with only 2 outcomes (predicts a probability between 0 and 1)

Logistic Regression should be used since we want to predict 1 of 2 possible outcomes: whether a user will convert or not depending on the page (old or new).

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 [358]:
```
df2['intercept'] = 1
df2[['new_page', 'old_page']] = pd.get_dummies(df2['landing_page'])
df2[['control', 'ab_page']] = pd.get_dummies(df2['group'])
df2.head()
```

```
/Users/karenbevis/anaconda3/lib/python3.6/site-packages/ipykernel/__mai
n__.py:1: 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-d
ocs/stable/indexing.html#indexing-view-versus-copy
  if __name__ == '__main__':
/Users/karenbevis/anaconda3/lib/python3.6/site-packages/pandas/core/fra
me.py:2540: 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-d
ocs/stable/indexing.html#indexing-view-versus-copy
  self[k1] = value[k2]
```

Out[358]:

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

View head to make sure all the new columns are accurate (0s and 1s are where they're supposed to be), then drop extraneous columns to simplify our df.

In [359]:
```
# only need ab_page (= treatment = new), old_page (= control), and converted
# group and landing_page have new dummy columns, can drop both but want to keep in just to check for accuracy
# drop one variable from 'group' dummies: control (since this is the same as old_page)
# drop one variable from 'landing_page' dummies: new_page (since this is the same as ab_page)

df2.drop(['new_page', 'control' ], axis=1, inplace=True)
df2.head()
```

```
/Users/karenbevis/anaconda3/lib/python3.6/site-packages/ipykernel/__main__.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#indexing-view-versus-copy
```

Out[359]:

	user_id	timestamp	group	landing_page	converted	intercept	old_page	ab_p
0	851104	2017-01-21 22:11:48.556739	control	old_page	0	1	1	0
1	804228	2017-01-12 08:01:45.159739	control	old_page	0	1	1	0
2	661590	2017-01-11 16:55:06.154213	treatment	new_page	0	1	0	1
3	853541	2017-01-08 18:28:03.143765	treatment	new_page	0	1	0	1
4	864975	2017-01-21 01:52:26.210827	control	old_page	1	1	1	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 [360]:
```
# I kept getting an error so ran this code:
from scipy import stats
stats.chisqprob = lambda chisq, df: stats.chi2.sf(chisq, df)
```

In [361]:
```
# use logistic model
logit_mod = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])
results = logit_mod.fit()
results.summary()
```

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

Out[361]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290584
Model:	Logit	**Df Residuals:**	290582
Method:	MLE	**Df Model:**	1
Date:	Wed, 16 May 2018	**Pseudo R-squ.:**	8.077e-06
Time:	15:31:36	**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

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

Logistic interpretation differs slightly from linear. We need to exponentiate the coefficients. The resulting value is the multiplicative change in the odds. If the coefficient is negative we take the reciprocal, and flip 'increase' to 'decrease' in the interpretation so it's easier to understand.

In [363]:
```
1/np.exp(-0.0150)
```

Out[363]: 1.015113064615719

Intrepretation: The new page is 1.015 times LESS likely to convert a user holding all else constant.

We use dummy variables to add categorical data to the matrix. When using dummies, we must drop one of the categories (since it is inferred by the remaining category/ categories), and the intercept becomes the baseline (dropped) category. Since we dropped 'control' and kept 'ab_page', it becomes our baseline:

intercept definition: predicted value of the response (y) variable when the (x) variable is zero
intercept = control group = old page (baseline): conversions are -1.9888
p-value for old_page: 0 (statistically significant, useful in predicting a response)

slope definition: expected change in response (y) variable for each 1 unit increase in the (x) variable
ab_page = treatment group = new page: -0.015 from intercept (less conversions than old page, negative slope)
p-value for ab_page: 0.190 (not statistically significant, not useful in predicting a response)

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**?

In regression, the p-value is given for testing if a parameter (for intercept or slope) = 0 (null) or does not = 0 (alternative).

Null: parameter = 0
Alternative: parameter != 0

In other words, is the variable useful for predicting a response? Smaller p-values (< 0.05) suggest a specific variable is statistically significant in relating to the response variable (the slope associated with (x) in relating to (y) is non-zero.)

p-value for old page (x): 0.0
Statistically significant in relating to the response variable (the slope associated with (x) in relating to (y) is non-zero)

p-value for new (ab) page (x): 0.19
Not statistically significant in relating to the response variable (the slope associated with (x) in relating to (y) is zero)

These p-values differ from Part II because in the A/B test our null hypothesis states that the old page is better than, or equal to, the new (a one-tailed test). Regression is a two-tailed test therefore p-values have a different meaning relating to slope.

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?

One disadvantage of adding additional terms into a regression model is **collinearity**. We want our predictor (X) variables to correlate with our response (y) variable (have a linear relationship), but when we add more predictor variables they might end up correlating with each other. We can use sb.pairplot to compare x variables, and VIF (variance inflation factors) to remove the least important variable if two variables are strongly related (VIF > 10).

Also, when (X) variables are not linearly related, **higher order terms** (such as interactions, quadratics, and cubics) might be considered. For example, if the slopes of two x variables are not parallel, this indicates that an interaction between those two variables is likely present. In this case, we can create a new column that multiplies these two values and review this added interaction with the predictor (y) variable.

Other considerations:

- **Change aversion**: gives unfair advantage to control group/ old page; users might be unhappy with change.
- **Novelty effect**: gives unfair advantage to treatment group/ new page; users might be drawn to change.

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 (https://pandas.pydata.org/pandas-docs/stable/generated/pandas.DataFrame.join.html) 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 [364]:
```
countries_df = pd.read_csv('./countries.csv')
df_new = countries_df.set_index('user_id').join(df2.set_index('user_id'
), how='inner')
df_new.head()
```

Out[364]:

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

In [365]:
```
# review country column data, how many unique entries are there?
df_new['country'].unique()
```

Out[365]:
```
array(['UK', 'US', 'CA'], dtype=object)
```

In [366]:
```
# Create the necessary dummy variables, list new cols in alpha order
# Kept all 3 dummies in the df to check for accuracy, will drop from lm

df_new[['CA', 'UK', 'US']] = pd.get_dummies(df_new['country'])
df_new.head()
```

Out[366]:

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

In [367]:
```
# drop US (now baseline)
logit_mod = sm.Logit(df_new['converted'], df_new[['intercept', 'CA', 'UK']])
results = logit_mod.fit()
results.summary()
```

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

Out[367]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290584
Model:	Logit	**Df Residuals:**	290581
Method:	MLE	**Df Model:**	2
Date:	Wed, 16 May 2018	**Pseudo R-squ.:**	1.521e-05
Time:	15:40:18	**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

In [368]:
```
1/np.exp(-0.0408), np.exp(0.0099)
```

Out[368]: (1.0416437559600236, 1.0099491671175422)

Compared to US overall conversions:

- CA is 1.04 times LESS likely to convert
- UK is 1.01 times MORE likely to convert

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.

Review combined (higher order) data by creating new columns with interactions between country and page.

In [369]:
```
df_new['UK_new_page'] = df_new['UK']*df_new['ab_page']
df_new['US_new_page'] = df_new['US']*df_new['ab_page']
df_new['CA_new_page'] = df_new['CA']*df_new['ab_page']
df_new.head()
```

Out[369]:

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

In [370]:
```
# drop US (now baseline)
logit_mod = sm.Logit(df_new['converted'], df_new[['intercept', 'CA_new_p
age', 'UK_new_page']])
results = logit_mod.fit()
results.summary()
```

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

Out[370]:

Logit Regression Results

Dep. Variable:	converted	**No. Observations:**	290584
Model:	Logit	**Df Residuals:**	290581
Method:	MLE	**Df Model:**	2
Date:	Wed, 16 May 2018	**Pseudo R-squ.:**	2.364e-05
Time:	15:41:48	**Log-Likelihood:**	-1.0639e+05
converged:	True	**LL-Null:**	-1.0639e+05
		LLR p-value:	0.08085

	coef	**std err**	**z**	**P>|z|**	**[0.025**	**0.975]**
intercept	-1.9963	0.006	-322.049	0.000	-2.008	-1.984
CA_new_page	-0.0752	0.038	-1.997	0.046	-0.149	-0.001
UK_new_page	0.0149	0.017	0.862	0.389	-0.019	0.049

In [371]:
```
1/np.exp(-0.0752), np.exp(0.0149)
```

Out[371]: (1.0780997492739288, 1.0150115583846535)

For the treatment group (those that received the new page) and compared to US conversions:

- CA is 1.078 times LESS likely to convert
- UK is 1.015 times MORE likely to convert

Compared to the US, CA conversions overall and for the new page is lower, and in the UK overall and new page conversions is higher. Perhaps CA has more change aversion and the UK is drawn to change, but these observed differences are quite small.

Conclusions

Congratulations on completing the project!

Gather Submission Materials

Once you are satisfied with the status of your Notebook, you should save it in a format that will make it easy for others to read. You can use the **File -> Download as -> HTML (.html)** menu to save your notebook as an .html file. If you are working locally and get an error about "No module name", then open a terminal and try installing the missing module using `pip install <module_name>` (don't include the "<" or ">" or any words following a period in the module name).

You will submit both your original Notebook and an HTML or PDF copy of the Notebook for review. There is no need for you to include any data files with your submission. If you made reference to other websites, books, and other resources to help you in solving tasks in the project, make sure that you document them. It is recommended that you either add a "Resources" section in a Markdown cell at the end of the Notebook report, or you can include a `readme.txt` file documenting your sources.

Submit the Project

When you're ready, click on the "Submit Project" button to go to the project submission page. You can submit your files as a .zip archive or you can link to a GitHub repository containing your project files. If you go with GitHub, note that your submission will be a snapshot of the linked repository at time of submission. It is recommended that you keep each project in a separate repository to avoid any potential confusion: if a reviewer gets multiple folders representing multiple projects, there might be confusion regarding what project is to be evaluated.

It can take us up to a week to grade the project, but in most cases it is much faster. You will get an email once your submission has been reviewed. If you are having any problems submitting your project or wish to check on the status of your submission, please email us at dataanalyst-project@udacity.com. In the meantime, you should feel free to continue on with your learning journey by beginning the next module in the program.