Analyze A/B Test Results

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

Specific programming tasks are marked with a **ToDo** tag.

Introduction

A/B tests are very commonly performed by data analysts and data scientists. For this project, you will be working to understand the results of an A/B test run by an e-commerce website. Your goal is to work through this notebook to help the company understand if they should:

- Implement the new webpage,
- Keep the old webpage, or
- Perhaps run the experiment longer to make their decision.

Each **ToDo** task below has an associated quiz present in the classroom. Though the classroom quizzes are **not necessary** to complete the project, they help ensure you are on the right track as you work through the project, and you can feel more confident in your final submission meeting the rubric (https://review.udacity.com/#!/rubrics/1214/view) specification.

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

Part I - Probability

To get started, let's import our libraries.

In [2]:
```
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
%matplotlib inline
#We are setting the seed to assure you get the same answers on quizzes as we set up
random.seed(42)
```

ToDo 1.1

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

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

In [3]:
```
#Read in the dataset from the `ab_data.csv` and desplay first rows
df = pd.read_csv('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

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

In [4]:
```
#num of rows inthe dataset
df.shape
```

Out[4]:
```
(294478, 5)
```

c. The number of unique users in the dataset.

In [5]:
```
#unique users in the dataset
df.nunique()
```

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

d. The proportion of users converted.

In [6]:
```
#the proportion of users converted
df.converted.mean()
```

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

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

In [7]:
```
#check the number of times when 'treatment' and 'new_page' don't line-up (two combinations)
df.query('group == "treatment" and landing_page != "new_page"') #first combination
```

Out[7]:

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

	user_id	timestamp	group	landing_page	converted
288465	767964	2017-01-19 09:41:32.875795	treatment	old_page	1
289242	698366	2017-01-04 00:22:43.306653	treatment	old_page	0
289665	693835	2017-01-20 11:44:50.517253	treatment	old_page	0
289799	909162	2017-01-09 17:12:38.910965	treatment	old_page	0
289846	934943	2017-01-04 18:45:15.921776	treatment	old_page	0
290062	928175	2017-01-05 03:51:08.933502	treatment	old_page	1
290149	858910	2017-01-10 05:20:37.997730	treatment	old_page	1
290328	658911	2017-01-05 15:14:40.331200	treatment	old_page	0
290360	714840	2017-01-10 23:35:22.559510	treatment	old_page	1
290647	904581	2017-01-17 11:35:54.031953	treatment	old_page	0
291313	807667	2017-01-15 19:11:59.976235	treatment	old_page	0
291754	795252	2017-01-19 02:43:07.343575	treatment	old_page	1
291922	634098	2017-01-07 23:45:07.976016	treatment	old_page	0
292412	693843	2017-01-09 06:31:48.749071	treatment	old_page	1
292521	689329	2017-01-06 03:58:15.546309	treatment	old_page	0
292607	699462	2017-01-17 23:54:08.826755	treatment	old_page	0
292800	712112	2017-01-14 23:33:41.083796	treatment	old_page	0
292963	742202	2017-01-12 04:34:20.344485	treatment	old_page	0
292977	638460	2017-01-22 13:38:30.677806	treatment	old_page	0
293240	861420	2017-01-04 20:34:09.065070	treatment	old_page	0
293302	825937	2017-01-04 20:56:48.825875	treatment	old_page	0
293391	934444	2017-01-12 19:49:35.581289	treatment	old_page	0
293443	738761	2017-01-04 15:20:52.694440	treatment	old_page	0
293530	934040	2017-01-04 20:52:26.981566	treatment	old_page	0
293773	688144	2017-01-16 20:34:50.450528	treatment	old_page	1
293817	876037	2017-01-17 16:15:08.957152	treatment	old_page	1
293917	738357	2017-01-05 15:37:55.729133	treatment	old_page	0
294014	813406	2017-01-09 06:25:33.223301	treatment	old_page	0
294252	892498	2017-01-22 01:11:10.463211	treatment	old_page	0

1965 rows × 5 columns

In [8]:
```
df.query ('group == "control" and landing_page != "old_page"') #second combination
##summed result from both combinations = 3893 = number of times when treatment and new_page don't match
```

Out[8]:

	user_id	timestamp	group	landing_page	converted
22	767017	2017-01-12 22:58:14.991443	control	new_page	0
240	733976	2017-01-11 15:11:16.407599	control	new_page	0
490	808613	2017-01-10 21:44:01.292755	control	new_page	0
846	637639	2017-01-11 23:09:52.682329	control	new_page	1
850	793580	2017-01-08 03:25:33.723712	control	new_page	1
988	698120	2017-01-22 07:09:37.540970	control	new_page	0
1198	646342	2017-01-06 18:39:23.484797	control	new_page	0
1354	735021	2017-01-16 09:51:29.349493	control	new_page	0
1474	678638	2017-01-18 06:36:42.515395	control	new_page	0
1877	717682	2017-01-07 03:05:39.891873	control	new_page	0
2023	937692	2017-01-19 01:29:42.739007	control	new_page	0
2214	649781	2017-01-20 03:50:20.837704	control	new_page	0
2745	872666	2017-01-05 07:44:32.050781	control	new_page	0
2759	639817	2017-01-06 23:39:11.754971	control	new_page	0
2857	738999	2017-01-08 15:21:55.309961	control	new_page	0
2947	847673	2017-01-07 18:45:04.253063	control	new_page	1
3362	858458	2017-01-06 04:51:33.183576	control	new_page	1
3421	638068	2017-01-20 01:57:00.012096	control	new_page	0
3548	807355	2017-01-21 11:10:28.793058	control	new_page	0
3817	832098	2017-01-15 06:06:26.163307	control	new_page	0
3903	855630	2017-01-10 16:24:01.119709	control	new_page	1
3913	937090	2017-01-22 07:38:49.397402	control	new_page	0
4038	919582	2017-01-04 12:24:28.755065	control	new_page	0
4282	866677	2017-01-24 05:04:14.004157	control	new_page	0
4284	847508	2017-01-03 19:31:14.396402	control	new_page	0
4311	924330	2017-01-23 07:08:56.964247	control	new_page	0
4485	838568	2017-01-15 04:02:13.337797	control	new_page	0
4693	799659	2017-01-22 09:50:16.421384	control	new_page	0
4748	872738	2017-01-08 02:16:03.976589	control	new_page	0
4962	729859	2017-01-19 14:17:09.976523	control	new_page	0
...	...	...	...	...	...
290811	931254	2017-01-19 03:56:48.943007	control	new_page	0

	user_id	timestamp	group	landing_page	converted
291093	922957	2017-01-12 00:58:45.303371	control	new_page	0
291100	810979	2017-01-07 18:48:46.403714	control	new_page	0
291240	807517	2017-01-22 10:07:39.903169	control	new_page	0
291358	929094	2017-01-11 03:52:10.013362	control	new_page	0
291423	848305	2017-01-19 07:30:03.635089	control	new_page	0
291728	828985	2017-01-02 13:55:08.790046	control	new_page	0
291839	740434	2017-01-13 07:04:11.067609	control	new_page	0
291876	766031	2017-01-03 22:49:27.025028	control	new_page	0
291946	861129	2017-01-12 19:00:59.118294	control	new_page	1
292147	746367	2017-01-10 04:37:37.933511	control	new_page	0
292178	645830	2017-01-14 11:12:33.289733	control	new_page	0
292235	679326	2017-01-07 07:27:46.910711	control	new_page	0
292239	908003	2017-01-22 15:17:03.083738	control	new_page	0
292405	819974	2017-01-03 05:58:44.734645	control	new_page	0
292570	778969	2017-01-21 12:59:42.740399	control	new_page	1
292748	684361	2017-01-19 03:59:57.656614	control	new_page	0
292845	893018	2017-01-10 15:05:37.522921	control	new_page	0
293017	792268	2017-01-06 09:21:58.341063	control	new_page	0
293085	884635	2017-01-19 14:19:48.484389	control	new_page	0
293393	636565	2017-01-12 07:26:31.103374	control	new_page	0
293480	638376	2017-01-18 15:41:02.395882	control	new_page	0
293568	704024	2017-01-15 17:06:09.309987	control	new_page	0
293662	927109	2017-01-04 09:14:33.647192	control	new_page	0
293888	865405	2017-01-12 08:38:50.511434	control	new_page	0
293894	741581	2017-01-09 20:49:03.391764	control	new_page	0
293996	942612	2017-01-08 13:52:28.182648	control	new_page	0
294200	928506	2017-01-13 21:32:10.491309	control	new_page	0
294253	886135	2017-01-06 12:49:20.509403	control	new_page	0
294331	689637	2017-01-13 11:34:28.339532	control	new_page	0

1928 rows × 5 columns

In [9]:
```
# to confirm the above
# 3893 = number of times when treatment and new_page don't line up
len(df.query("(group == 'control') and (landing_page == 'new_page')") + df.query("(group == 'treatment') and\
                                                                                  (landing_page == 'old_page')"))
```

Out[9]:
```
3893
```

f. Do any of the rows have missing values?

In [10]:
```
df.info() #check for null values
```

```
<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
```

ToDo 1.2

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

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

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

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

Store your new dataframe in **df2**.

In [11]:
```
# Remove the inaccurate rows, and store the result in a new dataframe df2
df2 = df.drop(df[((df.group == 'control') & (df.landing_page == 'new_page')) | \
                 ((df.group == 'treatment') & (df.landing_page == 'old_page'))].index)
```

In [12]:
```
#check the first rows of the new df2
df2.head()
```

Out[12]:

	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 [13]:
```
#check number of d2 rows and columns
df2.shape
```

Out[13]: (290585, 5)

In [14]:
```
df2.describe()
```

Out[14]:

	user_id	converted
count	290585.000000	290585.000000
mean	788004.825246	0.119597
std	91224.582639	0.324490
min	630000.000000	0.000000
25%	709035.000000	0.000000
50%	787995.000000	0.000000
75%	866956.000000	0.000000
max	945999.000000	1.000000

In [15]:
```
# Double Check all of the incorrect rows were removed from df2 -
# Output of the statement below should be 0
df2[((df2['group'] == 'treatment') == (df2['landing_page'] == 'new_page')) == False].shape[0]
```

Out[15]: 0

ToDo 1.3

Use **df2** and the cells below to answer questions for **Quiz 3** in the classroom.

a. How many unique **user_id**s are in **df2**?

In [16]:
```
#find unique users in df2
df2.user_id.nunique()
```

Out[16]:
```
290584
```

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

In [17]:
```
#check if there is a duplicated user_id in df2
sum(df2.user_id.duplicated())
```

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

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

In [18]:
```
## display the row information for the repeat user_id  = 773192
df2[df2.duplicated(['user_id'], keep=False)]
```

Out[18]:

	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

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

In [19]:
```
# Remove one of the rows with a duplicate user_id..

# the solution found on stackoverflow:
# https://stackoverflow.com/questions/13035764/remove-pandas-rows-with-duplicate-indices
df2 = df2[~df2.user_id.duplicated(keep = 'first')]

#second solution that also works:
#df2.drop_duplicates(['user_id'], inplace=True)

# Check again if the row with a duplicate user_id is deleted or not
#df2.shape
sum(df2.user_id.duplicated())
```

Out[19]:
```
0
```

ToDo 1.4

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

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

In [20]:
```
#the probability of an individual converting regardless of the page they receive
df2.converted.mean()
```

Out[20]: 0.11959708724499628

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

In [21]:
```
#Given that an individual was in the control group, what is the probability they converted
df2_control_group = df2.query('group == "control"')
df2_control_group.converted.mean()
```

Out[21]: 0.1203863045004612

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

In [22]:
```
#Given that an individual was in the treatment group, what is the probability they converted
df2_treatment_group = df2.query('group == "treatment"')
df2_treatment_group.converted.mean()
```

Out[22]: 0.11880806551510564

In [23]:
```
# Calculate the actual difference (obs_diff) between the conversion rates for the two groups.
# observed difference:
obs_diff = (df2_treatment_group.converted.mean()) - (df2_control_group.converted.mean())
obs_diff
```

Out[23]: -0.0015782389853555567

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

In [24]:
```
# the probability that an individual received the new page
len(df2_treatment_group.index)/len(df2.index)
```

Out[24]: 0.5000619442226688

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

-The treatment group has a conversion rate of 11.88%, while the control group has a conversion rate of 12.04%. -The observed difference in conversion rate is small: 0.16%. -The probability of conversion in the treatment group is slightly lower than in control group. Based on these results, we can assume that the treatment group will not lead to more conversions than the control group. It would need more investigation to check if the result is unbiased.

Part II - A/B Test

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

However, then the hard questions would be:

- Do you stop as soon as one page is considered significantly better than another or does it need to happen consistently for a certain amount of time?
- How long do you run to render a decision that neither page is better than another?

These questions are the difficult parts associated with A/B tests in general.

ToDo 2.1

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 be your null and alternative hypotheses (H_0 and H_1)?

You can state your hypothesis in terms of words or in terms of p_{old} and p_{new}, which are the "converted" probability (or rate) for the old and new pages respectively.

If we assume that the old page is better unless the new page proves to be definitely better at a Type I error rate of 5%, then:

-null hypothesis (H0)) is that the converted rate of the old page is greater or equal to the converted rate of the new page p_old >= p_new

-alternative hypothesis (H1)is that the converted rate of the new age is greater than the converted rate of the old page p_new > p_old

ToDo 2.2 - Null Hypothesis H_0 Testing

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

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

In this section, you will:

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

Use the cells below to provide the necessary parts of this simulation. You can use **Quiz 5** in the classroom to make sure you are on the right track.

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

In [25]:
```
#conversion rate for  p_new  under the null hypothesis
p_new = df2.converted.mean()
p_new
```

Out[25]: 0.11959708724499628

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

In [26]:
```
#conversion rate for p_old under the null hypothesis
p_old = df2.converted.mean()
p_old
```

Out[26]: 0.11959708724499628

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

In [27]:
```
#n_new the number of individuals in the treatment group
n_new = len(df2_treatment_group.index)
n_new
```

Out[27]: 145310

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

In [28]:
```
# n_old the number of individuals in the control group
n_old = len(df2_control_group.index)
print(n_old)
```

```
145274
```

e. Simulate Sample for the `treatment` Group

Simulate n_{new} transactions with a conversion rate of p_{new} under the null hypothesis.

Store these n_{new} 1's and 0's in the `new_page_converted` numpy array.

In [29]:
```
# Simulate a Sample for the treatment Group
new_page_converted = np.random.choice([1, 0], size = len(df2_treatment_group.index), p = [df2.converted.mean(), (1 - (df2.converted.mean()))])
```

f. Simulate Sample for the `control` Group

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

In [30]:
```
# Simulate a Sample for the control Group
old_page_converted = np.random.choice([1, 0], size=len(df2_control_group.index), p=[df2.converted.mean(), (1-(df2.converted.mean()))])
```

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

In [31]:
```
 #the difference in the "converted" probability  (p_new - p_old)
#for the above simulated samples
samples_p_diff = new_page_converted.mean() - old_page_converted.mean()
print(samples_p_diff)
```

```
0.00202155278002
```

h. Sampling distribution

Re-create `new_page_converted` and `old_page_converted` and find the $(p'_{new} - p'_{old})$ value 10,000 times using the same simulation process you used in parts (a) through (g) above.

Store all $(p'_{new} - p'_{old})$ values in a NumPy array called `p_diffs`.

In [32]:
```
# Sampling distribution
p_diffs = []
for _ in range(10000):
    new_page_converted = np.random.choice([1, 0], size=len(df2_treatment_group.index), p=[df2.converted.mean(), (1-(df2.converted.mean()))])
    old_page_converted = np.random.choice([1, 0], size=len(df2_control_group.index), p=[df2.converted.mean(), (1-(df2.converted.mean()))])
    p_diffs.append(new_page_converted.mean() - old_page_converted.mean())
```

In [33]:
```
#we can also do the same using different methods:
#old_page_converted= []
#new_page_converted = []
#p_diffs = []

# for loops are much slower than numpy functions
#for _ in range(10000):
 #   sample_old2 = df2_control_group.sample(n_old, replace=True)
 #   sample_new2 = df2_treatment_group.sample(n_new, replace=True)

 #   control_conversion = sample_old2['converted'].sum() / n_old
 #   treatment_conversion = sample_new2['converted'].sum() / n_new
```

In [34]:
```
# numpy binomial function would generate the distribution given that the null is true
#control_conversion = np.random.binomial(n_old, p_old, 10000) / n_old
#treatment_conversion = np.random.binomial(n_new, p_new, 10000) / n_new

#old_page_converted.append(control_conversion)
#new_page_converted.append(treatment_conversion)
#p_diffs.append(treatment_conversion - control_conversion)

#p_diffs = np.array(p_diffs)
```

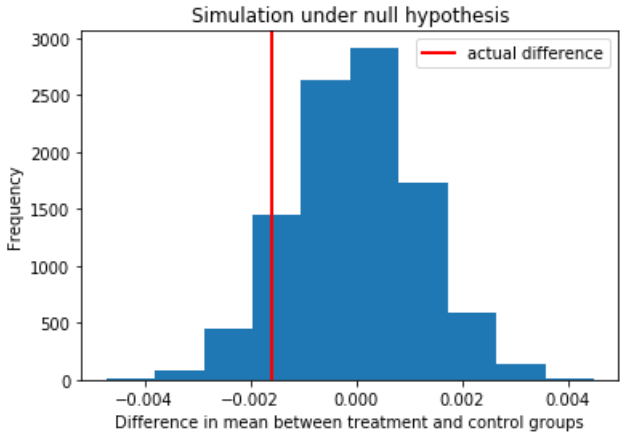
i. Histogram

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

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

In [35]:
```
obs_diff = (df2_treatment_group.converted.mean()) - (df2_control_group.converted.mean())
plt.hist(p_diffs);
plt.title('Simulation under null hypothesis')
plt.xlabel('Difference in mean between treatment and control groups')
plt.ylabel('Frequency')
plt.axvline(x = obs_diff, color = 'r', linewidth = 2, label = 'actual difference')
plt.legend()
```

Out[35]: <matplotlib.legend.Legend at 0x7f939cda0128>



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

In [36]:
```
p_value = (p_diffs > obs_diff).mean()
print(p_value)
```

```
0.9044
```

k. Please explain in words what you have just computed in part **j** above.

- What is this value called in scientific studies?
- What does this value signify in terms of whether or not there is a difference between the new and old pages? *Hint*: Compare the value above with the "Type I error rate (0.05)".

The proportion of the p_diffs that are greater than the actual difference observed in the df2 data is called the p-value.

The p-value is the probability of observing your statistic or - in other words - the probability of observing a difference as extreme as the one observed, if the null hypothesis is true.

Our null hypothesis was that the difference in means would be equal or less than 0. Our alternative hypothesis was that the difference would be greater than 0. Our difference is less then 0 though and the p-value of 0.9085 is large (with the significance level set at 0.05). We do not have evidence to reject the null hypotesis.

I. Using Built-in Methods for Hypothesis Testing

We could also use a built-in to achieve similar results. Though using the built-in might be easier to code, the above portions are a walkthrough of the ideas that are critical to correctly thinking about statistical significance.

Fill in the statements below to calculate the:

- `convert_old` : number of conversions with the old_page
- `convert_new` : number of conversions with the new_page
- `n_old` : number of individuals who were shown the old_page
- `n_new` : number of individuals who were shown the new_page

In [37]:
```
import statsmodels.api as sm


# number of conversions with the old_page
convert_old = len(df2_control_group[df2_control_group['converted'] == 1])
print(convert_old)

# number of conversions with the new_page
convert_new = len(df2_treatment_group[df2_treatment_group['converted'] == 1])
print(convert_new)

# number of individuals who were shown the old_page
n_old = len(df2_control_group.index)
print(n_old)

# number of individuals who received new_page
n_new = len(df2_treatment_group.index)
print(n_new)
```

```
/opt/conda/lib/python3.6/site-packages/statsmodels/compat/pandas.py:56: FutureWarning: The pandas.core.datetools module is deprecated and will be remov
ed in a future version. Please use the pandas.tseries module instead.
  from pandas.core import datetools

17489
17264
145274
145310
```

m. Now use `sm.stats.proportions_ztest()` to compute your test statistic and p-value. Here (https://www.statsmodels.org/stable/generated/statsmodels.stats.proportion.proportions_ztest.html) is a helpful link on using the built in.

The syntax is:

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

where,

- `count_array` = represents the number of "converted" for each group
- `nobs_array` = represents the total number of observations (rows) in each group
- `alternative` = choose one of the values from `['two-sided', 'smaller', 'larger']` depending upon two-tailed, left-tailed, or right-tailed respectively.

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

About the two-sample z-test

Recall that you have plotted a distribution `p_diffs` representing the difference in the "converted" probability $(p'_{new} - p'_{old})$ for your two simulated samples 10,000 times.

Another way for comparing the mean of two independent and normal distribution is a **two-sample z-test**. You can perform the Z-test to calculate the Z_score, as shown in the equation below:

$$Z_{score} = \frac{(p'_{new} - p'_{old}) - (p_{new} - p_{old})}{\sqrt{\frac{\sigma^2_{new}}{n_{new}} + \frac{\sigma^2_{old}}{n_{old}}}}$$

where,

- p' is the "converted" success rate in the sample
- p_{new} and p_{old} are the "converted" success rate for the two groups in the population.
- σ_{new} and σ_{new} are the standard deviation for the two groups in the population.
- n_{new} and n_{old} represent the size of the two groups or samples (it's same in our case)

> Z-test is performed when the sample size is large, and the population variance is known. The z-score represents the distance between the two "converted" success rates in terms of the standard error.

Next step is to make a decision to reject or fail to reject the null hypothesis based on comparing these two values:

- Z_{score}
- Z_{α} or $Z_{0.05}$, also known as critical value at 95% confidence interval. $Z_{0.05}$ is 1.645 for one-tailed tests, and 1.960 for two-tailed test. You can determine the Z_{α} from the z-table manually.

Decide if your hypothesis is either a two-tailed, left-tailed, or right-tailed test. Accordingly, reject OR fail to reject the null based on the comparison between Z_{score} and Z_{α}. We determine whether or not the Z_{score} lies in the "rejection region" in the distribution. In other words, a "rejection region" is an interval where the null hypothesis is rejected iff the Z_{score} lies in that region.

Reference:

- Example 9.1.2 on this page (https://stats.libretexts.org/Bookshelves/Introductory_Statistics/Book%3A_Introductory_Statistics_(Shafer_and_Zhang)/09%3A_Two-Sample_Problems/9.01%3A_Comparison_of_Two_Population_Means-_Large_Independent_Samples), courtesy www.stats.libretexts.org

In [38]:
```
import statsmodels.api as sm
# ToDo: Complete the sm.stats.proportions_ztest() method arguments
z_score, p_value = sm.stats.proportions_ztest([convert_old, convert_new], [n_old, n_new], alternative = 'smaller')
print(z_score, p_value)
```

```
1.31092419842 0.905058312759
```

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.**?

-Critical value at 95% confidence interval ($Z\alpha$) is 1.960 for two-tailed test (our test is a two tailed test as we are testing for the difference) what means that a z-score past -1.96 or 1.96 would be significant. The convertion rate of the new landing page is 1.3109 standard deviations from the convertion rate of the old landing page. 1.3109 is less than critical value of 1.96 so we failed to reject the null hypothesis.
-The p-value was calculated for null hypothesis that the new page would convert more than the old page. The alternative was that the old page converted more than or equal to the new page. The p-value of 0.9051 is much greater than a significance level of 0.05. so it supports the conclusion that we failed to reject the null hypothesis. The conclusion agrees with the findings from j.k.

Part III - A regression approach

ToDo 3.1

In this final part, you will see that the result you achieved in the A/B test in Part II above can also be achieved by performing regression.

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

We shoud perform logistic regression. "The logistic regression model estimates the probability of the binary outcome (conversion or no conversion) based on the values of the independent variables. It uses a logistic function to transform the linear regression equation into a range between 0 and 1, representing the probability of the binary outcome.

In the context of analyzing A/B test results, logistic regression can be used to determine if there is a significant difference in conversion rates between two groups (e.g., control group and treatment group)" (Udacity GPT)

b. The goal is to use **statsmodels** library to fit the regression model you specified in part **a.** above to see if there is a significant difference in conversion based on the page-type a customer receives. However, you first need to create the following two columns in the `df2` dataframe:

1. `intercept` - It should be `1` in the entire column.
2. `ab_page` - It's a dummy variable column, having a value `1` when an individual receives the **treatment**, otherwise `0`.

In [39]:
```
#add intercept and ab_column to the df2
df2['intercept'] = 1
df2['ab_page'] = pd.get_dummies(df2['group'])['treatment']
df2.head()
```

Out[39]:

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

In [40]:
```
#another way to do the same:
#df2['intercept'] = 1
#df2['ab_page'] = 0
#ab_page_index = df2[df2['group'] == 'treatment'].index
#df2.loc[ab_page_index, "ab_page"] = 1
#df2.head()
```

c. Use **statsmodels** to instantiate your regression model on the two columns you created in part (b). above, then fit the model to predict whether or not an individual converts.

In [41]:
```
lm = sm.Logit(df2['converted'], df2[['intercept', 'ab_page']])
results = lm.fit()
```

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

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

In [43]:
```
results.summary2()
```

Out[43]:

Model:	Logit	No. Iterations:	6.0000
Dependent Variable:	converted	Pseudo R-squared:	0.000
Date:	2023-08-16 15:34	AIC:	212780.3502
No. Observations:	290584	BIC:	212801.5095
Df Model:	1	Log-Likelihood:	-1.0639e+05
Df Residuals:	290582	LL-Null:	-1.0639e+05
Converged:	1.0000	Scale:	1.0000

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
intercept	-1.9888	0.0081	-246.6690	0.0000	-2.0046	-1.9730
ab_page	-0.0150	0.0114	-1.3109	0.1899	-0.0374	0.0074

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

1.The p-value associated with ab_page is 0.1899 With a p-value of 0.1899, and the significance level 0.05, since the p-value is greater than 0.05, we fail to reject the null hypothesis. This suggests that there is not enough evidence to conclude that the difference between the pages is statistically significant.

2.In Part II, the p-value was calculated where: -null hypothesis = new page would convert more than the old page, -alternative hypothesis = the old page converted more than or equal to the new page.

3.In Part III, we used variables and a linear model to calculate the p-value where: -the null hypothesis = the difference between the pages is equal to 0, -the alternative hypothesis = difference between the pages is greater or less than 0.

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?

It is a good idea to add other factors as they might enrich the result. For example: -by adding the type of platform that users used to enter the page, we can check if platform choice has an impact on conversion, -by adding the time of the day when users enter the page, we can check if the time has an influence on conversion. However, we should be careful while adding additional terms to our regression model, as some variables might affect others and disturb the result.

g. Adding countries

Now along with testing if the conversion rate changes for different pages, also add an effect based on which country a user lives in.

1. You will need to read in the **countries.csv** dataset and merge together your `df2` datasets on the appropriate rows. You call the resulting dataframe `df_merged` . Here (https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.join.html) are the docs for joining tables.
2. Does it appear that country had an impact on conversion? To answer this question, consider the three unique values, `['UK', 'US', 'CA']` , in the `country` column. Create dummy variables for these country columns.

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

In [44]:
```
# Read the countries.csv
df_countries = pd.read_csv('countries.csv')
```

In [45]:
```
# Join with the df2 dataframe
df_new = df_countries.set_index('user_id').join(df2.set_index('user_id'), how='inner')
df_new.head()
```

Out[45]:

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

In [46]:
```
# Create the necessary dummy variables
dummy_countries = pd.get_dummies(df_new['country'])
df3 = dummy_countries.join(df_new, how = 'inner')
df3.head()
```

Out[46]:

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

h. Fit your model and obtain the results

Though you have now looked at the individual factors of country and page on conversion, we would now like to look at an interaction between page and country to see if are there significant effects on conversion. **Create the necessary additional columns, and fit the new model.**

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

In [47]:
```
# check from which country comes the majority of customers
# we will use it as the reference
df3['country'].value_counts()
```

Out[47]:
```
US    203619
UK     72466
CA     14499
Name: country, dtype: int64
```

In [48]:
```
# Fit your model, and summarize the results
lm2 = sm.Logit(df3['converted'], df3[['intercept', 'UK', 'CA']])
results = lm2.fit()
results.summary2()
```

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

Out[48]:

Model:	Logit	No. Iterations:	6.0000
Dependent Variable:	converted	Pseudo R-squared:	0.000
Date:	2023-08-16 15:36	AIC:	212780.8333
No. Observations:	290584	BIC:	212812.5723
Df Model:	2	Log-Likelihood:	-1.0639e+05
Df Residuals:	290581	LL-Null:	-1.0639e+05
Converged:	1.0000	Scale:	1.0000

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
intercept	-1.9967	0.0068	-292.3145	0.0000	-2.0101	-1.9833
UK	0.0099	0.0133	0.7458	0.4558	-0.0161	0.0360
CA	-0.0408	0.0269	-1.5178	0.1291	-0.0935	0.0119

Conclusions

-The model above uses only the country as the explanatory variable. -The p-values are relatively large what means that country variable doesn't significantly affect the conversion rate (or - in other words - the conversion rate doesn't depend on which country the user comes from). -We failed to reject the null hypothesis

-We can check if adding 'ab_page' will change anything:

In [49]:
```
#check if adding another variable will change anything:
lm2 = sm.Logit(df3['converted'], df3[['intercept', 'ab_page', 'UK', 'CA']])
results = lm2.fit()
results.summary2()
```

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

Out[49]:

Model:	Logit	No. Iterations:	6.0000
Dependent Variable:	converted	Pseudo R-squared:	0.000
Date:	2023-08-16 15:36	AIC:	212781.1253
No. Observations:	290584	BIC:	212823.4439
Df Model:	3	Log-Likelihood:	-1.0639e+05
Df Residuals:	290580	LL-Null:	-1.0639e+05
Converged:	1.0000	Scale:	1.0000

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
intercept	-1.9893	0.0089	-223.7628	0.0000	-2.0067	-1.9718
ab_page	-0.0149	0.0114	-1.3069	0.1912	-0.0374	0.0075
UK	0.0099	0.0133	0.7433	0.4573	-0.0162	0.0359
CA	-0.0408	0.0269	-1.5161	0.1295	-0.0934	0.0119

Adding ab_page as a variable didn't change anything significantly. The correlation coefficient is small for UK and CA so the relationship between country and conversion doesn't seem to be strong.The p-value for ab_page is high - we still fail to reject the null hypothesis.

Let's check if adding other variables will make any difference (I used Udacity chat GPT to help me with adding new columns showing landing page based on country and ab_group)

In [50]:
```
UK_new_p = df3['ab_page'] * df3['UK']
df3['UK_new_p'] = UK_new_p
```

In [52]:
```
CA_new_p = df3['ab_page'] * df3['CA']
df3['CA_new_p'] = CA_new_p
df3.head()
```

Out[52]:

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

In [54]:
```
#fit the model and summarize the results
lm4 = sm.Logit(df3['converted'], df3[['intercept', 'ab_page', 'UK', 'CA', 'UK_new_p', 'CA_new_p']])
results = lm4.fit()
results.summary2()
```

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

Out[54]:

Model:	Logit	No. Iterations:	6.0000
Dependent Variable:	converted	Pseudo R-squared:	0.000
Date:	2023-08-16 15:39	AIC:	212782.6602
No. Observations:	290584	BIC:	212846.1381
Df Model:	5	Log-Likelihood:	-1.0639e+05
Df Residuals:	290578	LL-Null:	-1.0639e+05
Converged:	1.0000	Scale:	1.0000

	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
intercept	-1.9865	0.0096	-206.3440	0.0000	-2.0053	-1.9676
ab_page	-0.0206	0.0137	-1.5052	0.1323	-0.0473	0.0062
UK	-0.0057	0.0188	-0.3057	0.7598	-0.0426	0.0311
CA	-0.0175	0.0377	-0.4652	0.6418	-0.0914	0.0563
UK_new_p	0.0314	0.0266	1.1807	0.2377	-0.0207	0.0835
CA_new_p	-0.0469	0.0538	-0.8718	0.3833	-0.1523	0.0585

-The p-values are still large. -We failed to reject the null hyphothesis.

Conclusions: I conducted multiple tests to check if the new page will effectively increase conversion rates. All tests' results suggest that the new page won't increase conversion rate, I failed to reject the null hypothesis. That means that new page won't be better than the old page. It might be reasonable to conduct more tests (with more variables). However, for now - I would recommend keeping the old page.

Final Check!

Congratulations! You have reached the end of the A/B Test Results project! You should be very proud of all you have accomplished!

Submission

You may either submit your notebook through the "SUBMIT PROJECT" button at the bottom of this workspace, or you may work from your local machine and submit on the last page of this project lesson.

1. Before you submit your project, you need to create a .html or .pdf version of this notebook in the workspace here. To do that, run the code cell below. If it worked correctly, you should get a return code of 0, and you should see the generated .html file in the workspace directory (click on the orange Jupyter icon in the upper left).

1. Alternatively, you can download this report as .html via the **File** > **Download as** submenu, and then manually upload it into the workspace directory by clicking on the orange Jupyter icon in the upper left, then using the Upload button.

1. Once you've done this, you can submit your project by clicking on the "Submit Project" button in the lower right here. This will create and submit a zip file with this .ipynb doc and the .html or .pdf version you created. Congratulations!

In [65]:
```
from subprocess import call
call(['python', '-m', 'nbconvert', 'Analyze_ab_test_results_notebook.ipynb'])
```

Out[65]: 0