Loading Librarys

In [7]:
```
import pandas as pd
import numpy as np
import random
import matplotlib.pyplot as plt
%matplotlib inline
random.seed(55)
```

Reading Data

In [21]:
```
df =pd.read_csv(r'C:\Users\suchitra\Desktop\python\data\ab_data.csv')
```

In [22]:
```
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
```

In [23]:
```
df.head()
```

Out[23]:

	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 [24]:
```
df.groupby(['group','converted']).agg('count')
```

Out[24]:

group	converted	user_id	timestamp	landing_page
control	0	129479	129479	129479
	1	17723	17723	17723
treatment	0	129762	129762	129762
	1	17514	17514	17514

In [25]:
```
df.drop(df.query("group == 'control' and landing_page == 'new_page'").index, inplace=True)
df.drop(df.query("group == 'treatment' and landing_page == 'old_page'").index, inplace=True)

df.groupby(['group','converted']).agg('count')
```

Out[25]:

group	converted	user_id	timestamp	landing_page
control	0	127785	127785	127785
	1	17489	17489	17489
treatment	0	128047	128047	128047
	1	17264	17264	17264

In [26]:
```
df.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
```

Cleaning Duplicated Users

In [27]:
```
df[df.duplicated(['user_id'], keep=False)]
```

Out[27]:

	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

In [28]:
```
df.drop_duplicates(['user_id'], inplace=True)
assert len(df['user_id'].unique()) == df['user_id'].size
df.info()
```

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

In [29]:
```
df['converted'].mean()
```

Out[29]: 0.11959708724499628

In [30]:
```
df.groupby(['group']).describe()
```

Out[30]:

	user_id							
	count	mean	std	min	25%	50%	75%	max
group								
control	145274.0	788164.072594	91287.914601	630002.0	709279.50	788128.5	867208.25	945
treatment	145310.0	787845.719290	91161.564429	630000.0	708745.75	787876.0	866718.75	945

In [31]:
```
df.groupby(['group']).agg({'converted' : ['sum', 'count', 'mean']})
```

Out[31]:

	converted		
	sum	count	mean
group			
control	17489	145274	0.120386
treatment	17264	145310	0.118808

In [32]:
```
df[['group', 'converted']].groupby(['group']).agg('mean')
```

Out[32]:

	converted
group	
control	0.120386
treatment	0.118808

In [33]:
```
df[['group', 'converted']].groupby(['group']).agg('mean').T
```

Out[33]:

group	control	treatment
converted	0.120386	0.118808

Calculate Prob. of New and Old Pages Respectively

In [34]:
```
p_old_page = df[['group', 'converted']].query("group == 'control'")['converte
d'].mean()
p_new_page = df[['group', 'converted']].query("group == 'treatment'")['convert
ed'].mean()
act_p_diff = p_new_page - p_old_page

print('p_old_page:\t{}\np_new_page:\t{}\np_diff:\t\t{}'.format(p_old_page, p_n
ew_page, act_p_diff))
```

```
p_old_page:     0.1203863045004612
p_new_page:     0.11880806551510564
p_diff:         -0.0015782389853555567
```

Calculate Counts of New and Old Pages Respectively

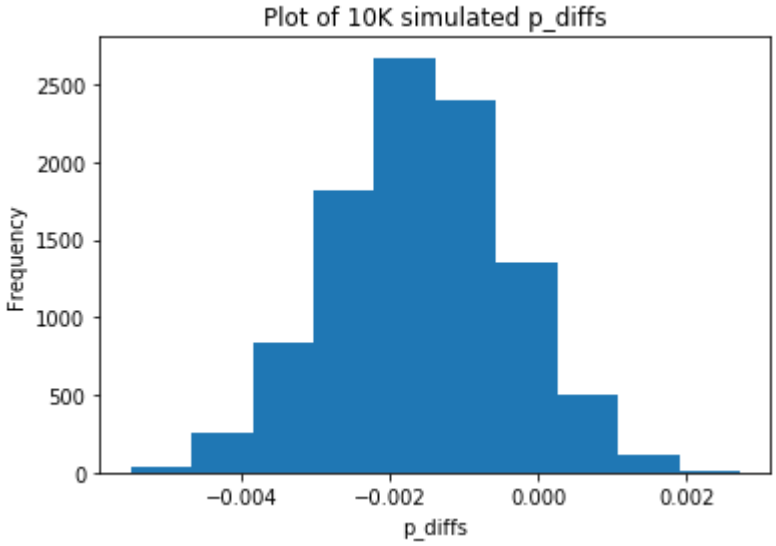
In [35]:
```
n_old =  len(df[['group']].query("group == 'control'"))
n_new =  len(df[['group']].query("group == 'treatment'"))

print('n_old:\t{}\nn_new:\t{}'.format(n_old, n_new))
```

```
n_old:  145274
n_new:  145310
```

Simulating Randomly With Respect to Probs.

In [39]:
```
p_diffs=[]
for _ in range(10000):
    new_page_converted = np.random.choice([1, 0], size=n_new, p=[p_new_page, (
1 - p_new_page)]).mean()
    old_page_converted = np.random.choice([1, 0], size=n_old, p=[p_old_page, (
1 - p_old_page)]).mean()
    diff = new_page_converted - old_page_converted
    p_diffs.append(diff)
```

In [40]:
```
plt.hist(p_diffs)
plt.xlabel('p_diffs')
plt.ylabel('Frequency')
plt.title('Plot of 10K simulated p_diffs');
```



In [41]:
```
p_diffs = np.array(p_diffs)
(act_p_diff < p_diffs).mean()
```

Out[41]: 0.4996