This Analysis covers hypothesis formulation, testing, and interpreation. Data is received from Kaggle (https://www.kaggle.com/zhangluyuan/ab-testing?select=ab_data.csv) - created to test the effectivness of 2 designs of a website page.

Here are the 5 steps of this Analysis:

```
##### 1. Experiment Definition
##### 2. Data collection and preparation
##### 3. Visualization
##### 4. Hypothesis testing
##### 5. Conclusions
```

1. EXPERIMENT DEFINITION

The data is related with an online e-commerce business, they are doing some changes in the website and want to see if the new website helps to convert customer to purchaser. Current conversion rate is 13%, if there is an increase 2%, then the effectivness of new webpage will be justified. So, with new website 15% conversion rate needs to be created. So, we are trying to test if the new webpage is better than the old one, but we don't know either it is better or worst. So, we are doing tow-tailed test:

H0: p = p0

H1: p != p0

p & p0 are the conversion rate of the new and old design. Our confidence level is 95%: a = 0.05

a: “if the probability of observing a result as extreme or more (p-value) is lower than α, then we reject the Null hypothesis”. a = 0.05 means 5% probability with confidence of (1 - a) of 95%.

Define variables

Control group - group of customers shown the old design.

Treatment group - group of customers shown the new design.

Our dependent variable (i.e., our target measurement) is the conversion_rate, which is a binary variable: 0 means user did not buy the product and 1 means user bought the product.

sample size for each group is calculated from the "Power Analysis", which depends on: (a) Power of test (1-B) -- probability of finding a statistical difference between the groups in test set when a difference is actually present. This is usually set at 0.8 by convention. (b) Alpha value (a) - The critical value set at 0.05. (c) Effect size -- How big of a difference we expect there to be between teh conversion rates.

```
    ##### we need a difference of 2%, so we need to use 13% and 15% to
calculate the effect size we want. Rest Python will caclculate.
```

In [2]:

```python
# Packages imports
import numpy as np
import pandas as pd
```

```
import scipy.stats as stats
import statsmodels.stats.api as sms
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
from math import ceil

%matplotlib inline
```

In [3]:

```
# plot styling preferences
plt.style.use('seaborn-whitegrid')
font = {'family': 'Helvetica', 'weight':'bold', 'size': 14}
mpl.rc('font', **font)

# effect size calculation based on our expected rates
effect_size = sms.proportion_effectsize(0.13, 0.15)

# calculating sample size needed
required_n = sms.NormalIndPower().solve_power(effect_size, power=0.8, alpha=0.05, ratio

# Rounding up to next whole number
required_n = ceil(required_n)

print(required_n)
```

```
4720
```

Above result suggests, we need at least 4720 observations for each group.

The meaning of power = 0.8 suggests if actual difference in conversion rates (13% vs. 15%) exists between our designs, there is about 80% chance to detect it statistically significant in our test with the sample size we calculated.

2. Data collection and preparation

data is downloaded from Kaggle. Python is used to prepare and analyze the data. Each group will consists of 4720 random rows.

In [5]:

```
df = pd.read_csv('ab_data.csv')
df.head()
```

Out[5]:

	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 [6]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 294478 entries, 0 to 294477
Data columns (total 5 columns):
 #   Column        Non-Null Count   Dtype 
---  ------        --------------   ----- 
 0   user_id       294478 non-null  int64 
 1   timestamp     294478 non-null  object
 2   group         294478 non-null  object
 3   landing_page  294478 non-null  object
 4   converted     294478 non-null  int64 
dtypes: int64(2), object(3)
memory usage: 11.2+ MB
```

In [7]:

```
## To make sure all the control group are seeing the old page and viceversa

pd.crosstab(df['group'], df['landing_page'])
```

Out[7]:

landing_page	new_page	old_page
group		
control	1928	145274
treatment	145311	1965

For this analysis we only need the group and converted columns. But, before sampling, let's make sure there are no users that have been sampled multiple times.

In [8]:

```
session_counts = df['user_id'].value_counts(ascending = False)
multi_users = session_counts[session_counts > 1].count()

print(f'There are {multi_users} that appear multiple times in the dataset')
```

```
There are 3894 that appear multiple times in the dataset
```

In [10]:

```
# Lets remove the repetition from the DataFrame to avoid samping the same users twice.

users_to_drop = session_counts[session_counts > 1].index

df = df[~df['user_id'].isin(users_to_drop)]
print(f'The updated dataset now has {df.shape[0]} entries')
```

```
The updated dataset now has 286690 entries
```

In [11]:

```
# Lets sample our data from each group using DataFrame.sample(), which performs Simple
# random_state = 22 suggests the results are reproducible.

control_sample = df[df['group'] == 'control'].sample(n=required_n, random_state = 22)
treatment_sample = df[df['group'] == 'treatment'].sample(n=required_n, random_state = 2

ab_test = pd.concat([control_sample, treatment_sample], axis = 0)
ab_test.reset_index(drop = True, inplace = True)

ab_test
```

Out[11]:

	user_id	timestamp	group	landing_page	converted

	user_id	timestamp	group	landing_page	converted
0	763854	2017-01-21 03:43:17.188315	control	old_page	0
1	690555	2017-01-18 06:38:13.079449	control	old_page	0
2	861520	2017-01-06 21:13:40.044766	control	old_page	0
3	630778	2017-01-05 16:42:36.995204	control	old_page	0
4	656634	2017-01-04 15:31:21.676130	control	old_page	0
...	...	...	...	...	...
9435	908512	2017-01-14 22:02:29.922674	treatment	new_page	0
9436	873211	2017-01-05 00:57:16.167151	treatment	new_page	0
9437	631276	2017-01-20 18:56:58.167809	treatment	new_page	0
9438	662301	2017-01-03 08:10:57.768806	treatment	new_page	0
9439	944623	2017-01-19 10:56:01.648653	treatment	new_page	1

9440 rows × 5 columns

In [12]:

```
ab_test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9440 entries, 0 to 9439
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   user_id       9440 non-null   int64
 1   timestamp     9440 non-null   object
 2   group         9440 non-null   object
 3   landing_page  9440 non-null   object
 4   converted     9440 non-null   int64
dtypes: int64(2), object(3)
memory usage: 368.9+ KB
```

In [13]:

```
ab_test['group'].value_counts()
```

Out[13]:

```
control      4720
treatment    4720
Name: group, dtype: int64
```

3. Visualising the results

In [14]:

```
conversion_rates = ab_test.groupby('group')['converted']

# Std. deviation of the proportion
std_p = lambda x: np.std(x, ddof = 0)

# Std. error of the proportion (std / sqrt(n))
se_p = lambda x: stats.sem(x, ddof = 0)

conversion_rates = conversion_rates.agg([np.mean, std_p, se_p])
conversion_rates.columns = ['conversion_rate', 'std_deviation', 'std_error']
```

```
conversion_rates.style.format('{:.3f}')
```

Out[14]:

group	conversion_rate	std_deviation	std_error
control	0.123	0.329	0.005
treatment	0.126	0.331	0.005

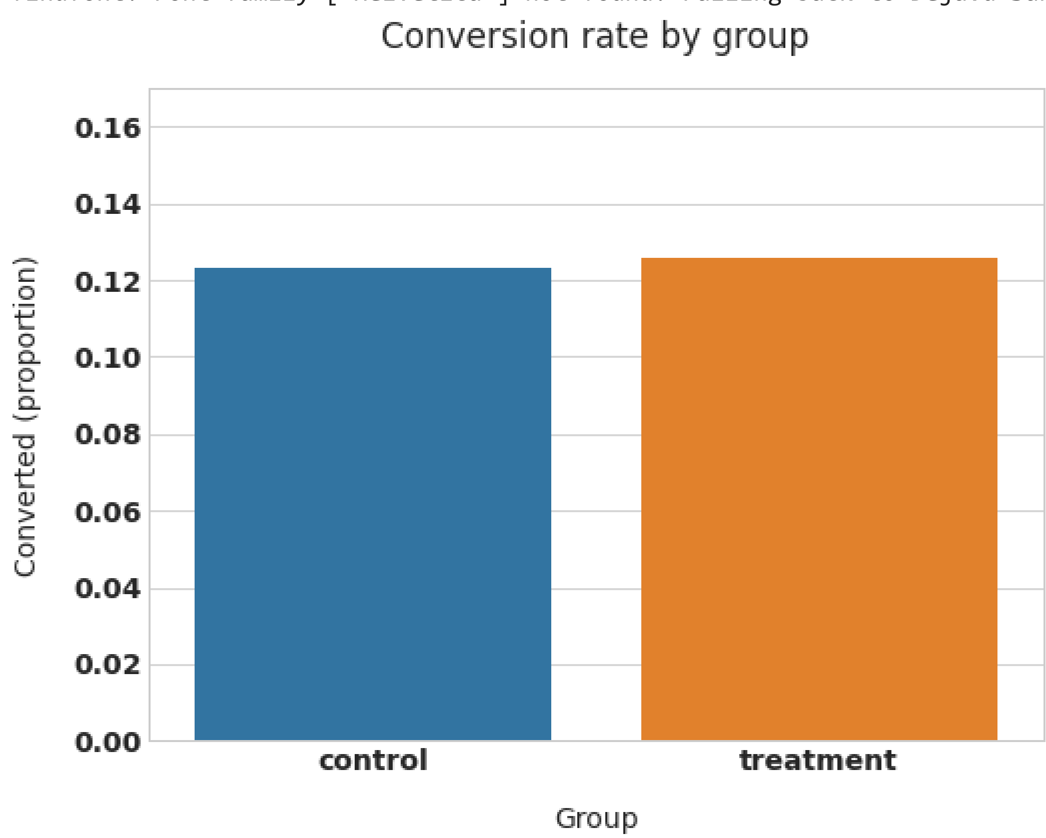
Above table shows conversion_rate for both group is close 12.3% vs 12.6%.

In [15]:

```
plt.figure(figsize = (8,6))

sns.barplot(x = ab_test['group'], y = ab_test['converted'], ci = False)

plt.ylim(0, 0.17)
plt.title('Conversion rate by group', pad=20)
plt.xlabel('Group', labelpad=15)
plt.ylabel('Converted (proportion)', labelpad=15)
```

Out[15]:

```
Text(0, 0.5, 'Converted (proportion)')

findfont: Font family ['Helvetica'] not found. Falling back to DejaVu Sans.
findfont: Font family ['Helvetica'] not found. Falling back to DejaVu Sans.
findfont: Font family ['Helvetica'] not found. Falling back to DejaVu Sans.
```



conversion rate of control group is slightly lower than treatment group. but, conversion rate of the control group is even lower than the average rate (12.3% vs. 13%). This might suggests that there is some variation in results when sampling from a population.

lets check if the treatment group's higher value is statistically significant.

4. Hypothesis testing

Due to large sample size, we can use the normal approximation for the calculation of the p-value (i.e., z-test).

we can use the statsmodels.stats.proportion module to get the p-value and confidence intervals.

In [16]:
```
from statsmodels.stats.proportion import proportions_ztest, proportion_confint

control_results = ab_test[ab_test['group'] == 'control']['converted']
treatment_results = ab_test[ab_test['group'] == 'treatment']['converted']

n_con = control_results.count()
n_treat = treatment_results.count()
successes = [control_results.sum(), treatment_results.sum()]
nobs = [n_con, n_treat]

z_stat, pval = proportions_ztest(successes, nobs=nobs)
(lower_con, lower_treat), (upper_con, upper_treat) = proportion_confint(successes, nobs

print(f'z statistic: {z_stat:.2f}')
print(f'p-value: {pval:.3f}')
print(f'ci 95% for control group: [{lower_con:.3f}, {upper_con:.3f}]')
print(f'ci 95% for treatment group: [{lower_treat:.3f}, {upper_treat:.3f}]')
```

```
z statistic: -0.34
p-value: 0.732
ci 95% for control group: [0.114, 0.133]
ci 95% for treatment group: [0.116, 0.135]
```

5. Conclusions

p-value of 0.732 suggests our result is statistically insignificant, that means we cannot reject the Null Hypothesis H0, which suggests that the new design did not perform significantly different than the old one.

Additionally, confidence interval for the treatment group is of 11.6% - 13.5%, which means it includes the baseline value of 13% conversion rate, but doesn't include target value of 15% (the 2% uplift).

Above result suggests that the new design doesn't increase the conversion rate, so new design is not an improvement over the old one.