A/B Testing

References:
https://github.com/Alicelibinguo/Analyzing-Website-Landing-Page-A-B-Test-Results-/blob/master/Analyze_ab_test_results_notebook.py

```
experiment_df = df2.query('group=="experiment"')
experiment_df.completed.mean()    # can use mean() directly for proportion because the values are true or false
```

2 hypothesis:
Null hypothesis
Alternative hypothesis

5 metrics :
Click through Rate
Enrollment Rate
Average Reading Duration
Average Classroom Time
Completion Rate

Experiment 1 background :
The first change Audacity wants to try is on their homepage. They hope that this new, more engaging design will increase the number of users that explore their courses, that is, move on to the second stage of the funnel.

The metric we will use is the click through rate for the Explore Courses button on the home page. **Click through rate (CTR)** is often defined as the the number of clicks divided by the number of views.
CTR = # clicks by unique users / # views by unique users.

```
import pandas as pd
df = pd.read_csv('homepage_actions.csv')
df.head()
```

	timestamp	id	group	action
0	2016-09-24 17:42:27.839496	804196	experiment	view
1	2016-09-24 19:19:03.542569	434745	experiment	view
2	2016-09-24 19:36:00.944135	507599	experiment	view
3	2016-09-24 19:59:02.646620	671993	control	view
4	2016-09-24 20:26:14.466886	536734	experiment	view

1. Match the following characteristics of this dataset:

- total number of actions
- number of unique users
- sizes of the control and experiment groups (i.e., the number of unique users in each group)

total number of actions
df['action'].value_counts().sum()   or df.shape

number of unique users
df['id'].nunique()   or   df.nunique()

# size of control group and experimental group
df.groupby('group').nunique()   or   df.group.value_counts()

	timestamp	id	group	action
group				
control	4264	3332	1	2
experiment	3924	2996	1	2

2. How long was the experiment run for?

Hint: the records in this dataset are ordered by timestamp in increasing order

# duration of this experiment
df.timestamp.max(), df.timestamp.min()

3. What action types are recorded in this dataset?

(i.e., What are the unique values in the action column?)

```
# action types in this experiment
df['action'].value_counts()     or     df.action.value_counts()
```

Output:
view   6328
click   1860
Name: action, dtype: int64

4. Why would we use click through rate (CTR) instead of number of clicks to compare the performances of control and experiment pages?

Two groups may have different sizes of visitors . So higher percentage of clicks does not mean more clicks over the other version.

Getting the proportion of the users who click is more effective than getting the number of users who click when comparing groups of different sizes.

5. Define the click through rate (CTR) for this experiment

The number of unique visitors who clicks at least once divided by the number of unique visitors who view the page.

Experiment 1 : should we implement the new homepage ?

Let's recap the steps we took to analyze the results of this A/B test.

1. We computed the **observed difference** between the metric, click through rate, for the control and experimental group.
2. We simulated the **sampling distribution** for the difference in proportions (or difference in click through rates).
3. We used this sampling distribution to simulate the **distribution under the null** hypothesis, by creating a random normal distribution centered at 0 with the same spread and size.

4. We computed the **p-value** by finding the proportion of values in the null distribution that were greater than our observed difference.
5. We used this p-value to determine the **statistical significance** of our observed difference.

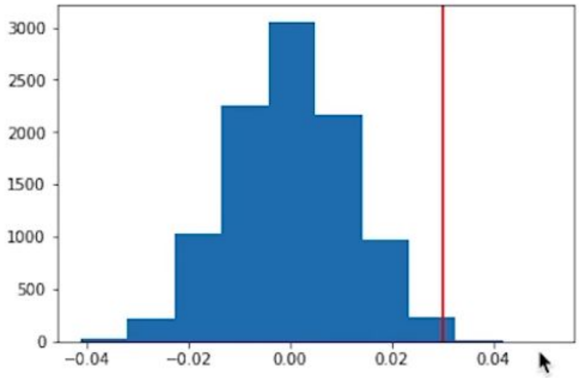
CTR

```
import pandas as pd
Import numpy as np
Import matplot.pyplot as plt
% matplotlib inline
df = pd.read_csv('homepage_actions.csv')
df.head()

##############   step 1:  observed difference in the sample ##############
# Control group Click through rate:
Control_df = df.query(‘ group == “control” ‘)
Control_ctr = Control_df.query(‘action==”click” ’).id.nunique()  /
Control_df.query(‘action==”view” ’).id.nunique()

# experiment group click through rate:
expe_df = df.query(‘ group == “control” ‘)   # double query
expe_ctr = expe_df.query(‘action==”click” ’).id.nunique()  / expe_df.query(‘action==”view”
’).id.nunique()

# the difference between this two groups
Obs_diff = Expe_ctr - Control_ctr

############## step 2: bootstrap the sample to simulate the sample distribution for the
difference in ctr ##################################
# Check if the difference is significant or by chance

diffs = []
for _ in range(10000):
    b_sample = df.sample( df.shape[0] , replace = True )

    control_df = b_sample.query(‘ group ==”control” ‘ )
    control_ctr = Control_df.query(‘action==”click” ’).id.nunique()  /
control_df.query(‘action==”view” ’).id.nunique()

    expe_df =  b_sample.query(‘ group ==”experiment” ‘ )
   expe_ctr = expe_df.query(‘action==”click” ’).id.nunique()  / expe_df.query(‘action==”view”
’).id.nunique()
```

diff.append(expe_ctr - Control_ctr)

plt.hist(diffs)

################## step 3: use the sample distribution to simulate the distribution under the null hypothesis ############################
# create a random normal distribution with central at 0 with same spread and size
diffs = np.array(diffs)
null_vals = np.random.normal(0 , diffs.std() , diffs.size)
plt.hist(null_vals)
plt.axvline(x = obs_diff , color = 'red')



################# step 4: find the p value #############
(Null_vals > obs_diff).mean()
output : 0.0053 < 0.05
 # A low p-value can give us a statistical evidence to support rejecting the null hypothesis,
CONCLUSION : smaller p, then alternative, which means statistically significant results. reject the null, adopt the new version.

Experiment 2 background:

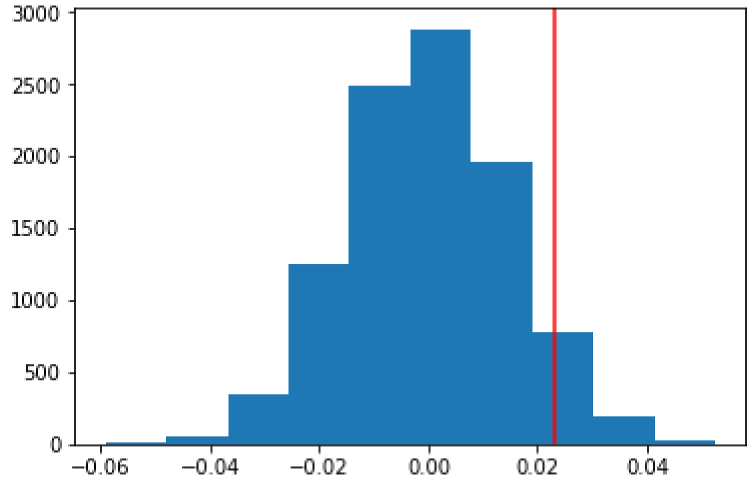
Enrollment Rate
Average Reading Duration
Average Classroom Time
Completion Rate

Experiment 2: Should we adopt the new course overview page ?

We need to check multiple rates (4) here to decide
############## metric 1 : Enrollment Rate ###############
import numpy as np
import pandas as pd

```
import matplotlib.pyplot as plt
% matplotlib inline

np.random.seed(42)

df = pd.read_csv('course_page_actions.csv')
df.head()

# Get dataframe with all records from control group
control_df = df.query('group == "control"')
# Compute enroll rate for control group
control_er = control_df.query('action == "enroll"').count()[0] / control_df.query('action ==
"view"').count()[0]
# Display enroll rate
Control_er

# Get dataframe with all records from experiment group
experiment_df = df.query( 'group=="experiment"')
# Compute enroll rate for experiment group
experiment_er = experiment_df.query('action == "enroll"').count()[0] /
experiment_df.query('action == "view"').count()[0]
# Display enroll rate
Experiment_er

# Compute the observed difference in enroll rates
obs_diff =  experiment_er - control_er
# Display observed difference
obs_diff

# Create a sampling distribution of the difference in proportions
# with bootstrapping
diffs = []
size = df.shape[0]
for _ in range(10000):
   b_samp = df.sample(size, replace=True)
   control_df = b_samp.query('group == "control"')
   control_ctr = control_df.query('action == "enroll"').count()[0] / control_df.query('action ==
"view"').count()[0]
   experiment_df = b_samp.query('group == "experiment"')   # double query
   experiment_ctr = experiment_df.query('action == "enroll"').count()[0] /
experiment_df.query('action == "view"').count()[0]
   diffs.append(experiment_ctr - control_ctr)

# Convert to numpy array
diffs = np.array(diffs)
```

```
# Plot sampling distribution
plt.hist(diffs)
# Simulate distribution under the null hypothesis
null_vals = np.random.normal( 0 , diffs.std() , diffs.size )

# Plot the null distribution
plt.hist( null_vals )

# Plot observed statistic with the null distribution
plt.hist(null_vals);
plt.axvline(obs_diff, c='red')
```



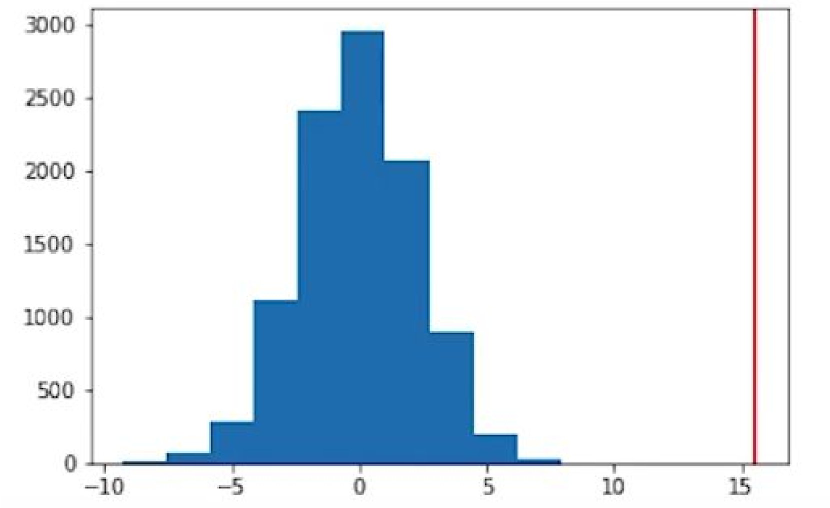

```
# Compute p-value
(null_vals > obs_diff).mean()
```

Output : 0.061 > 0.05

CONCLUSION : if a type I error rate of 0.05. P greater , then null . Fails to reject the null. No statistically significant results.

```
##############    metric 2 : Average Reading Duration    ###############
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
% matplotlib inline

np.random.seed(42)

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

```
# here the time duration only related to view
Views = df.query(‘ action==”view” ‘ )
Reading_times = views.groupby( [ “id” ,”group”]) [“duration”].mean()

# reset the index to keep the id and group as column names . Also, we can work on
dataframe instead of multi-series
reading_times = reading_times.reset_index()
reading_times.head()

control_mean = df.query(‘ group == “control” ‘ ) [‘duration’].mean()
experiment_mean = df.query(‘ group == “experiment” ‘ ) [‘duration’].mean()

Obs_diff =  experiment_mean - control_mean

# The following steps are the same with bootstrapping

# Create a sampling distribution of the difference in proportions
# with bootstrapping
diffs = []
size = df.shape[0]
for _ in range(10000):
    b_samp = df.sample(size, replace=True)
    control_mean = b_samp.query(‘ group == “control” ‘ ) [‘duration’].mean()
    experiment_mean = b_samp.query(‘ group == “experiment” ‘ ) [‘duration’].mean()
    diffs.append(experiment_mean - control_mean)

# Convert to numpy array
diffs = np.array(diffs)

# Plot sampling distribution
plt.hist(diffs)
# Simulate distribution under the null hypothesis
null_vals = np.random.normal( 0 , diffs.std() , diffs.size )

# Plot the null distribution
plt.hist( null_vals )

# Plot observed statistic with the null distribution
plt.hist(null_vals);
plt.axvline(obs_diff, c='red')
```

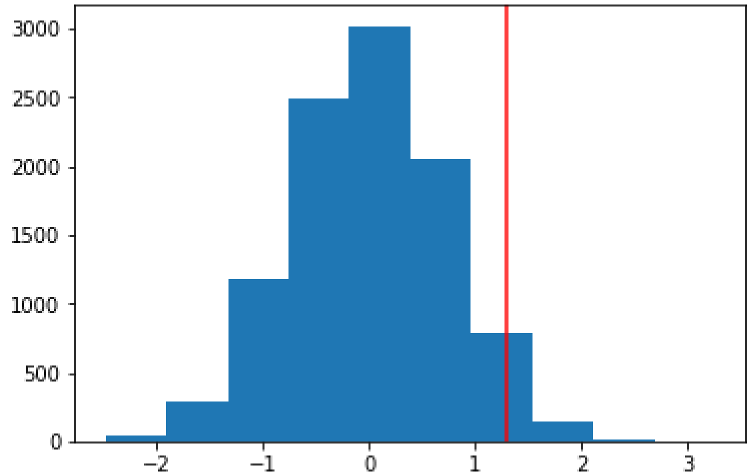



```
# Compute p-value
(null_vals > obs_diff).mean()
```

CONCLUSION : Observed statistics falls outside of the null distribution The difference we observed is significant. Fails to reject the null . P greater, then null. No statistically significant results.

```
###############    metric 3 : Average Classroom Time   ################
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
% matplotlib inline

np.random.seed(42)
df = pd.read_csv('classroom_actions.csv')
df.head()

# The total_days represents the total amount of time
# each student has spent in classroom.
# get the average classroom time for control group
control_mean = df.query('group == "control"')['total_days'].mean()

# get the average classroom time for experiment group
experiment_mean = df.query('group == "experiment"')['total_days'].mean()

# display average classroom time for each group
control_mean, experiment_mean
```

```
# compute observed difference in classroom time
obs_diff = experiment_mean - control_mean

# display observed difference
obs_diff

# create sampling distribution of difference in average classroom times
# with boostrapping
diffs = []
for _ in range(10000):
    boot_sample = df.sample(df.shape[0] , replace=True)
    control_mean = boot_sample.query('group == "control"')['total_days'].mean()
    experiment_mean = boot_sample.query('group == "experiment"')['total_days'].mean()
    diffs.append(experiment_mean - control_mean )

# convert to numpy array
diffs = np.array(diffs)

# plot sampling distribution
plt.hist(diffs)

# simulate distribution under the null hypothesis
# create a random normal distribution with central at 0 with same spread and size
null_vals = np.random.normal(0,diffs.std(),diffs.size)

# plot null distribution
plt.hist(null_vals)

# plot line for observed statistic
plt.axvline( x= obs_diff , color = 'red')

# compute p value
(null_vals>obs_diff).mean()
```



Output: 0.038399999999999997 < 0.05

CONCLUSION: P-value is less than type one error 0.05. P smaller, then alternative. statistically significant. But engaging students for 1.3 more days in the classroom, when they average around 74 days in total, doesn't seem to indicate a large enough value to launch this change from a practical perspective for Audacity.

```
##############    metric 4 : Completion Rate  ###############
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
% matplotlib inline

np.random.seed(42)
df = pd.read_csv('classroom_actions.csv')
df.head()

# Create dataframe with all control records
control_df = df.query('group=="control"')

# Compute completion rate
#control_cr = control_df.query('completed=="True"').id.nunique() / df.id.nunique()
control_cr = control_df.completed.mean()

# Display completion rate
control_cr
# Create dataframe with all experiment records
experiment_df = df.query('group=="experiment"')

# Compute completion rate
experiment_cr = experiment_df.completed.mean()

# Display completion rate
experiment_cr
# Compute observed difference in completion rates
obs_diff = experiment_cr - control_cr
```

```
# Display observed difference in completion rates
obs_diff
# Create sampling distribution for difference in completion rates
# with boostrapping
diffs = []
for _ in range(10000):
    b_sample = df.sample(df.shape[0],replace=True)
    control_df = b_sample.query('group=="control"')
    control_cr = control_df.completed.mean()
    experiment_df = b_sample.query('group=="experiment"')
    experiment_cr = experiment_df.completed.mean()
    diffs.append(experiment_cr - control_cr)

# convert to numpy array
diffs = np.array(diffs)
# plot distribution
plt.hist(diffs)
# create distribution under the null hypothesis
# create a normal distribution with center at 0 , same spreed and size with sample distribution
null_vals = np.random.normal(0, diffs.std() , diffs.size)
# plot null distribution
plt.hist(null_vals)

# plot line for observed statistic
plt.axvline(x = obs_diff , color = 'red')
# compute p value
(null_vals>obs_diff).mean()
```

Output: 0.084599999999999995 > 0.05

CONCLUSION: p greater, then null, no statistically significant results.

############## metric 5 : Analyze Multiple Metrics ###############

The more metrics you evaluate, the more likely you are to observe significant differences just by chance - similar to what you saw in previous lessons with multiple tests. Luckily, this multiple comparisons problem can be handled in several ways.

Bonferroni Correction is one way we could handle experiments with multiple tests, or metrics in this case. To compute the new bonferroni correct alpha value, we need to **divide the original alpha value by the number of tests.**