Tutorial 3: Estimation and Confidence Intervals

In [45]:
```
%reset
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import scipy as stats
from scipy import stats
import statsmodels.api as sm
import statsmodels as sm
```

Confidence Intervals

The ballistic coefficient is a measure of an object's ability to overcome air resistance in flight. That parameter is inversely proportional to the deceleration of a flying body and is very important for bullet proof personal equipment. The ballistic coefficient was measured for the bullets of two versions of 9 mm Makarov cartridges, PM and PMM (which is a later and modified version). Sample bullets are chosen randomly.

Is there evidence to support the claim that PMM cartridge types have different ballistic coefficients than PM types? Use α=0.05.

PM	63	57	58	62	66	58	61	60	55	62	59	60	58	
PMM	69	65	59	62	61	57	59	60	60	62	61	66	68	66

Part 1

Load in and explore the data: Create graphs and summary statistics to compare versiosn.

You can use pandas to create Series data and then concat to a dataframe.

Or import from .csv file.

In [46]:
```
A=pd.Series([63.0,57.0,58.0,62.0,66.0,58.0,61.0,60.0,55.0,62.0,59.0,60.0,58.
B=pd.Series([69.0,65.0,59.0,62.0,61.0,57.0,59.0,60.0,60.0,62.0,61.0,66.0,68.
df=pd.concat([A,B],axis="columns")
df
```

Out[46]:

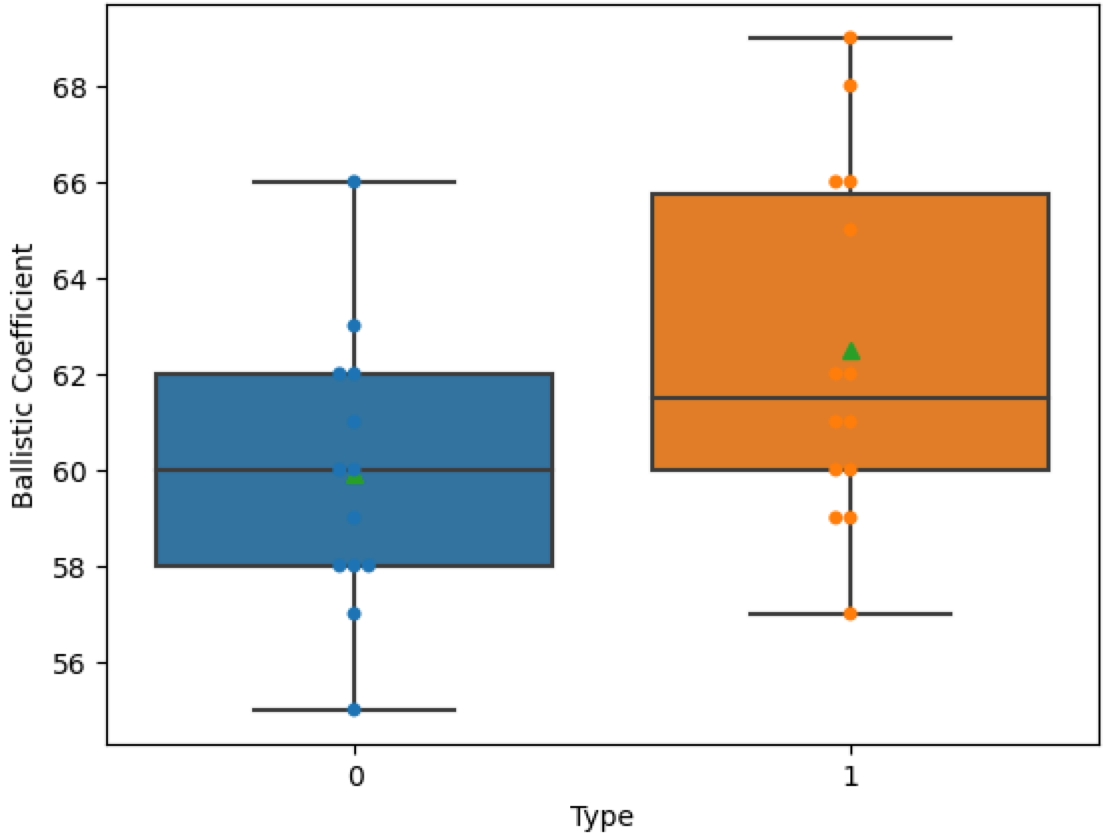
	PM	PMM
0	63.0	69.0
1	57.0	65.0
2	58.0	59.0
3	62.0	62.0
4	66.0	61.0
5	58.0	57.0
6	61.0	59.0
7	60.0	60.0
8	55.0	60.0
9	62.0	62.0
10	59.0	61.0
11	60.0	66.0
12	58.0	68.0
13	NaN	66.0

In [47]:

```
# 1. Plot the data to visualize
ax=sns.boxplot(data=[A, B], showmeans=True)
ax=sns.swarmplot(data=[A, B])
ax.set(xlabel="Type")
ax.set(ylabel="Ballistic Coefficient")
```

Out[47]:

```
[Text(46.972222222222214, 0.5, 'Ballistic Coefficient')]
```



In [48]:
```
# sumamry statistics fo Type A and Type B
df.describe()
```

Out[48]:

	PM	PMM
count	13.000000	14.000000
mean	59.923077	62.500000
std	2.900044	3.674235
min	55.000000	57.000000
25%	58.000000	60.000000
50%	60.000000	61.500000
75%	62.000000	65.750000
max	66.000000	69.000000

In [49]:
```
np.std(B, ddof=1)
```

Out[49]:
```
3.6742346141747673
```

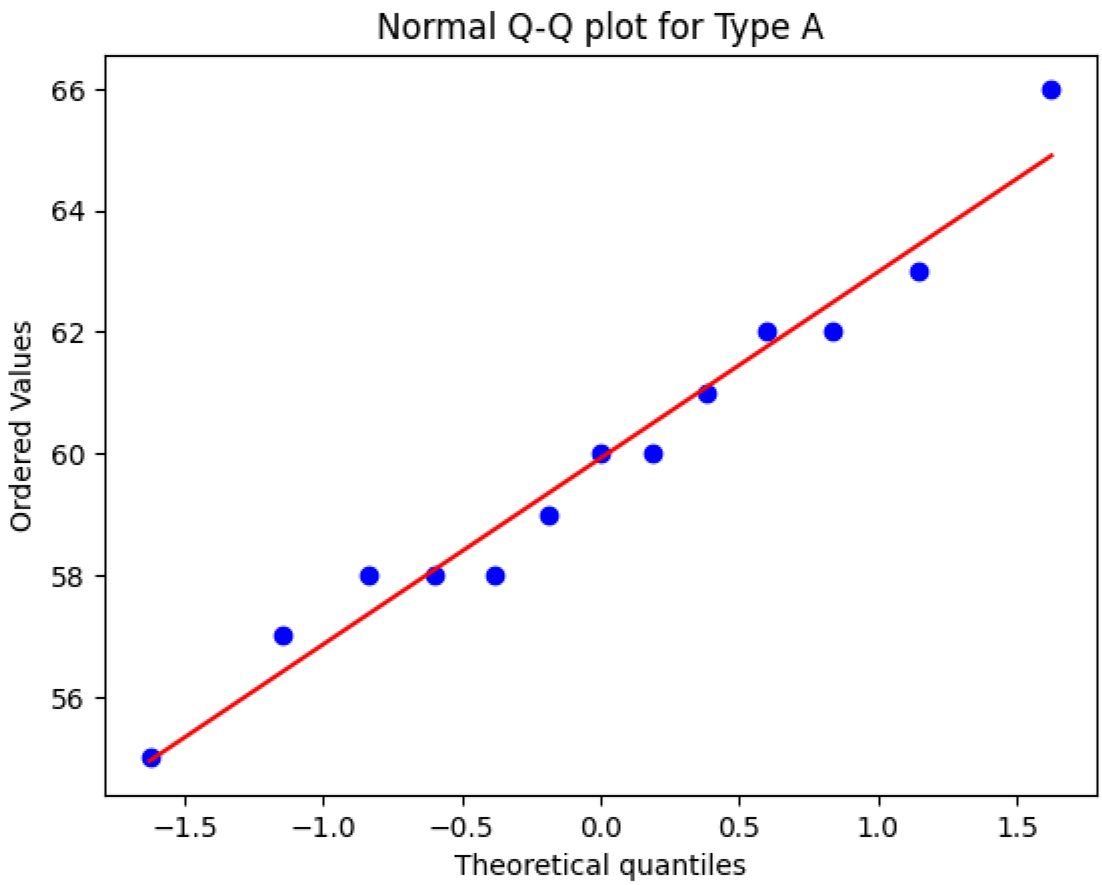
Part 2

We want to create 95% confidence intervals for the mean values of the different types?

Review the assumptions needed. Review and defend the assumption of normality for the data for the types.

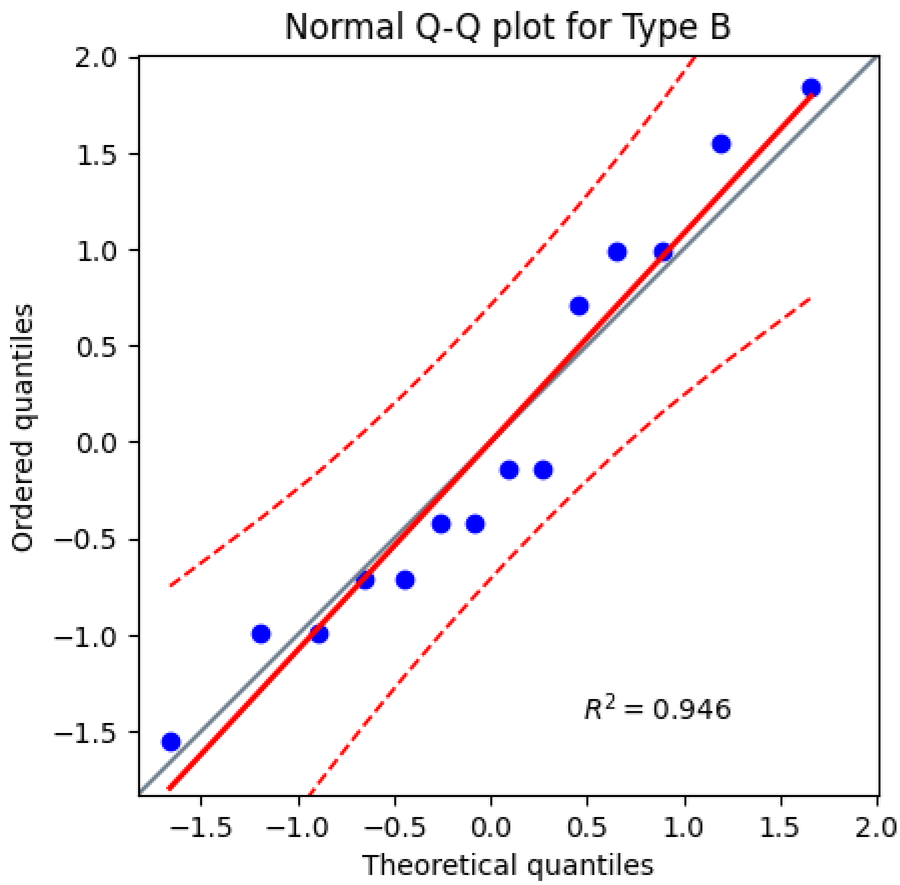
Let's look at the normal QQ plots

In [50]:
```
# normal probability plot using scipy models probplot
stats.probplot(A, dist="norm", plot=plt)
plt.title("Normal Q-Q plot for Type A")
plt.show()
```



In [51]:
```
import pingouin as pg

ax = pg.qqplot(df["PMM"], dist='norm', confidence = 0.95)
plt.title("Normal Q-Q plot for Type B")
plt.show()
```



Have you shown that the two samples are normally distributed

Part 3

Create individual 95% CI for the two groups of differnt types of bullets. Do this by hand and using a t-table and by python.

In [52]:
```
nA= df["PM"].count() # number in type A
nB= df["PMM"].count() # number in type B
print("The number of observations in the samples are: " + str(nA)+" and " +s
print("The standard deviation of the sampes are: "+ str(np.std(A,ddof=1))+"
print("The mean of the samples are: "+ str(np.mean(A)) +" and "+str(np.mean(
print("The variance of the samples are: "+str(np.std(A,ddof=1)**2)+" and "+
```

```
The number of observations in the samples are: 13 and 14
The standard deviation of the sampes are: 2.900044208327937 and 3.6742346141
747673
The mean of the samples are: 59.92307692307692 and 62.5
The variance of the samples are: 8.41025641025641 and 3.6742346141747673
```

In [53]:
```
# careful using len call using the dataframe; can use .count()
len(A)
```

Out[53]: 13

In [54]:
```
len(df["PM"])
```
Out[54]: 14

In [55]:
```
df["PM"].count()
```
Out[55]: 13

In [56]:
```
# Find SEM for the two groups
semA= np.std(A,ddof=1) / np.sqrt(nA)
semB= np.std(B,ddof=1) / np.sqrt(nB)
print(f"SEM_A: {semA}")
print(f"SEM_B: {semB}")
```
```
SEM_A: 0.804327545710673
SEM_B: 0.9819805060619657
```

In [57]:
```
# Find the 95% CI for the two groups
CI_A = stats.t.interval(0.95, df=nA-1, loc=np.mean(A), scale=semA)
CI_B = stats.t.interval(0.95, df=nB-1, loc=np.mean(B), scale=semB)
print(f"95% CI, A: {CI_A[0]} - {CI_A[1]}")
print(f"95% CI, B: {CI_B[0]} - {CI_B[1]}")
```
```
95% CI, A: 58.170597747230815 - 61.675556098923025
95% CI, B: 60.37856009344801 - 64.621439906552
```

Confidence Interval on the Mean, Variance Unknown

If $\bar{x}$ and s are the mean and standard deviation of a random sample from a normal distribution with unknown variance σ^2, a $100(1-\alpha)\%$ confidence interval on μ is given by

$$\bar{x} - t_{\alpha/2,n-1}s/\sqrt{n} \leq \mu \leq \bar{x} + t_{\alpha/2,n-1}s/\sqrt{n} \tag{8.16}$$

where $t_{\alpha/2,n-1}$ is the upper $100\alpha/2$ percentage point of the t distribution with $n-1$ degrees of freedom.

In [58]:
```
# critcial t-value for 0.025 and n-1
t_crit = stats.t.ppf(1-0.025, nA-1)
print(t_crit)
```
```
2.1788128296634177
```

In [59]:
```
# lower bound for type a
lowerA= np.mean(A) - t_crit*semA
print(lowerA)
```
```
58.170597747230815
```

In [60]:
```
# upper bound for type a
upperA= np.mean(A) + t_crit*semA
print(upperA)
```
```
61.675556098923025
```

Use SciPy stats Confidence interval using the t- distirbution
https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.t.html

verify using Python

note: for the stats.t.interval, df=degrees of freedom

loc=mean

scale=standard error of the mean

stats.t.interval(confidence= , df =, loc= , scale=)

In [72]:
```
stats.t.interval(confidence=0.95,df=((df["PM"].count())-1), loc=np.mean(df['
```

Out[72]:
```
(58.170597747230815, 61.675556098923025)
```

Part 4

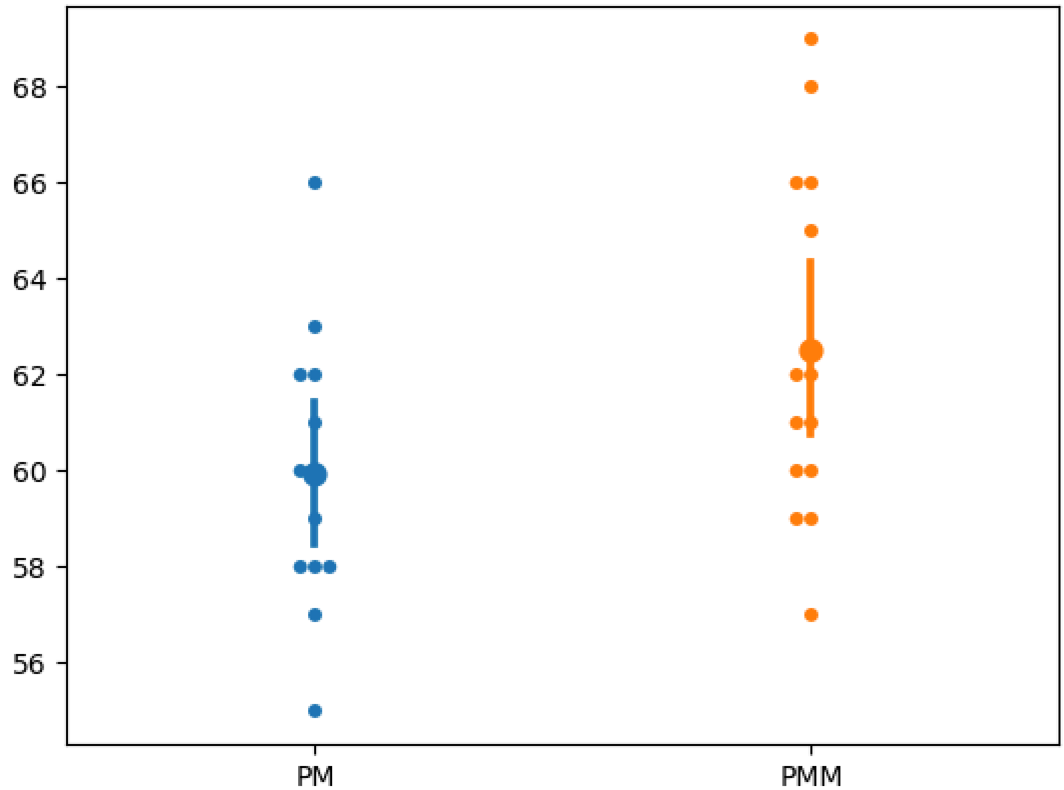
Graph the Means and Individual 95% Confidence Intervals for the groups.

You cat use seaborn catplot with kind=point to do this easily.

Or seaborn pointplot.

https://seaborn.pydata.org/generated/seaborn.catplot.html
https://seaborn.pydata.org/generated/seaborn.pointplot.html

In [62]:
```
fig, ax = plt.subplots(sharex=True, sharey=True)
sns.pointplot(data=df, estimator='mean', errorbar=('ci', 95), ax=ax, linesty
sns.swarmplot(data=df, ax=ax)
plt.show()
```



Part 5

Create individual 85% CI for the groups and provide a graph with the results. You can just use only python to calculate these and graph the results.

In [63]:
```
CI_A = stats.t.interval(0.85, df=nA-1, loc=np.mean(A), scale=semA)
CI_B = stats.t.interval(0.85, df=nB-1, loc=np.mean(B), scale=semB)
```

In [64]:
```
print(f"85% CI, A: {CI_A[0]} - {CI_A[1]}")
print(f"85% CI, B: {CI_B[0]} - {CI_B[1]}")
```

```
85% CI, A: 58.686056150421734 - 61.160097695732105
85% CI, B: 60.99764877082488 - 64.00235122917512
```

In [68]:
```
t_crit = stats.t.ppf(1-0.075, nA-1)
lowerA= np.mean(A) - t_crit*semA
upperA= np.mean(A) + t_crit*semA
print(lowerA, upperA)
```

```
58.686056150421734 61.160097695732105
```

In [69]:
```
t_crit = stats.t.ppf(1-0.075, nB-1)
lowerB= np.mean(B) - t_crit*semB
upperB= np.mean(B) + t_crit*semB
print(lowerB, upperB)
```

```
60.99764877082488 64.00235122917512
```

In [67]:
```
fig, ax = plt.subplots(sharex=True, sharey=True)
sns.pointplot(data=df, estimator='mean', errorbar=('ci', 85), ax=ax, linesty
sns.swarmplot(data=df, ax=ax)
plt.show()
```